UNIVERSITY OF SOUTHAMPTON

How Micro-Evolution Can Guide Macro-Evolution: Multi-Scale Search via Evolved Modular Variation

by

Rob Mills

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

April 2010

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Rob Mills

A divide-and-conquer approach to problem solving can in principle be far more efficient than tackling a problem as a monolithic whole. This type of approach is most appropriate when problems have the type of modular organisation known as near-decomposability, as implicit in many natural and engineered systems. Existing methods create higher-scale composite units from non-random combinations of lower-scale units that reflect sub-problem optima. The use of composite units affords search at a higher scale that, when applied recursively, can ultimately lead to optimal top-level solutions. But for this approach to be efficient, we must decompose a problem in a manner that respects its intrinsic modular structure, information which is in general unavailable *a priori*. Thus, identifying and subsequently exploiting the structure recursively is vital in providing fully automatic problem decomposition.

In this thesis, we define a family of algorithms that probabilistically adapt the scale of decomposition they use to reflect the structure in a problem. By doing so, they can provide optimisation that is provably superior to any single scale of search in nearly-decomposable problems. Our proposed framework couples two adaptive processes: a rapid, fine-scale search that guides a slower adaptation of the decomposition. This results in a scaling up of the units used in the rapid search, now operating at a macro-scale. We find that separating the timescales for the fine-scale search and the adaptation of the decomposition is crucial for this kind of scalable optimisation.

Using a simple and general class of problems that have no systematic structure, we demonstrate how our approach can nevertheless exploit the incidental structure present. Furthermore, we use idealised cases that have simple modular structure to demonstrate how our method scales as $\Theta(N \log N)$ (where N is the problem size), despite the fact that single-scale search methods scale as $\Omega\big(2^{\sqrt{N}}\big)$ – and support this distinction analytically.

Although our approach is algorithmically superior to single-scale search, the underlying principles that it is constructed from are simple and can operate using only localised feedback. We discuss intriguing parallels between our approach and the significance of associative evolution for ecosystem adaptation. Our results suggest that macro-evolutionary processes might not be merely extended micro-evolution, but that the action of evolutionary processes upon several scales is fundamentally different from the conventional view of (micro-)evolution at a single scale.

Contents

List of Figures

List of Tables

Acknowledgements

To those from whom I have learned, received direction, feedback, support, and scientific stimulation, I am indebted. Richard, Nic, Chris, Simon, Pat, Kathy, Jason, Devin, Molly, Laura, Josh, Alex, Simon, Ingo, Jeff, Tom, Seth, Oanh, Edmund, Adam, Tom, Andrew, Carrie, Ed, Elisabeth, Johannes, Frazer, Max: this thesis could not have happened without you.

To Coop

In loving memory

Nomenclature

ASIC	Application-specific integrated circuit
BBH	The Building block hypothesis
BBHC	Building block hill climber (algorithm)
BBO	Black-box optimisation
BMDA	Bivariate marginal distribution algorithm
BOA	Bayesian optimisation algorithm
CCEA	Cooperative coevolutionary algorithm
COMIT	Combining mutual information from trees (algorithm)
DEUM	Distribution estimation using Markov random fields (algorithm)
DSM	Dependency structure matrix
DSMGA	Dependency structure matrix genetic algorithm
EA	Evolutionary algorithm
EBNA	Estimation of Bayesian network algorithm
ECGA	Extended compact genetic algorithm
EDA	Estimation of distribution algorithm
ETA	Evolutionary transitions algorithm
FDA	Factorised distribution algorithm
GA	Genetic algorithm
GRA	Genetic renormalisation algorithm
hBOA	Hierarchical Bayesian optimisation algorithm
HGA	Hierarchical genetic algorithm
HIFF	Hierarchical if-and-only-if (problem)
HM-C	Hopfield network, used as a control (algorithm)
HM-EI	Hopfield network with evolved interactions (algorithm)
HXOR	Hierarchical exclusive-or (problem)
IFF	If-and-only-if (logical operator)
K2	A heuristic metric used for scoring Bayesian networks
LFDA	Learning factorised distribution algorithm
MA	Memetic algorithm
MACRO-H	Implementation of the MACRO framework, with ‘hard’ joins
MACRO-S	Implementation of the MACRO framework, with ‘soft’ joins
MACRO-D	Distributed Implementation of the MACRO framework

MIMIC	Mutual information maximising input clustering algorithm
NK	A problem landscape with tuneable epistasis
NKC	A pair of coupled NK-landscapes
NP	Non-deterministic polynomial
PBIL	Population-based incremental learning (algorithm)
RRB	rewired random-block (problem)
RSSA	Reciprocal synergy symbiosis algorithm
SBB	Scalable building blocks (problem)
SBB-N	Scalable building blocks with neutrality (problem)
SEAM	Symbiogenic evolutionary adaptation model
STAGE	An optimisation algorithm
TSP	Travelling salesperson problem
UMDA	Univariate marginal distribution algorithm
VSM	Variable structural modularity (problem)
XOR	Exclusive-or (logical operator)

Chapter 1

Introduction

Many of the problems that we face in the world today are large and complex. Solving such problems as a monolithic whole can be extremely difficult, but intuitively we can improve efficiency by splitting up a problem into smaller sub-problems. Each sub-problem can then be solved in partial isolation. Once solved, we then assemble a compatible collection of sub-solutions to produce a high quality top-level solution. At this higher level of search, the internal details of lower-level solutions are not important. However, there are often multiple solutions to any particular sub-problem, and the best choice depends on how the other sub-problems are solved. In short, the optimal solution for any one sub-problem is context sensitive (Watson and Pollack, 2005).

The notion that problem decomposition can offer greater efficiency is based on the tenet that smaller problems are radically easier to solve than larger problems – simple combinatorics confirm this situation – provided that the sub-problems are appropriately identified to avoid significant interactions. Here lies a central issue: for this approach to work, it is essential that a system is appropriately decomposed. Such information requires expert knowledge of the system under consideration, which is in general not available. How to render efficient problem solving under such 'knowledge-lean' conditions is an open question. One broad approach to address this is the development of methods that automatically discover a modular decomposition in the course of solving a problem.

Modularity is likely to make a problem or system more amenable to a problem decomposition approach and is a property that many engineered and biological systems possess in some form (Simon, 1969). For example, a car is made of many components, each of which could be designed with several variants (a large, powerful engine, or a compact and low capacity engine; a light transmission system suitable for a city car, or a heavy-duty transmission better suited to a pick-up), and those components can be designed largely, but not completely, in isolation from one another. The transmission system designer should take account of the expected torque output of the engine, but need not understand the internal details of how that power is achieved. Furthermore, each component could be broken down into smaller sub-components, with similar constraints. However, the design of products such as this is done in a top-down manner, with a system architect making decisions regarding the desired top-level functionality, and hence the requirements of each component, sub-component, and so on. A top-down approach is only really applicable for systems for which we have complete control. Note that this is unlike large problems that we face, such as design of the internet, or intervention in either the human body or the biosphere.

Conversely, in biological systems there is no such central architect. But this does not preclude the presence of modularity – far from it. Modularity has been observed in biological systems across many scales (Callebaut, 2005; Schlosser, 2002). When the effects of one genetic change only influence a subset of other genes, it can avoid the "generative entrenchment" that may otherwise cause almost all variation to be strongly deleterious (Wimsatt and Schank, 2004). Consolidated constraints are implicated in evolvability (Wagner and Altenberg, 1996); modular organisation is exhibited by genetic regulation networks (Hartwell et al., 1999; Ihmels et al., 2002) and ecosystems (Levin, 1999; Borrett et al., 2007; Krause et al., 2003).

The notion that engineers could borrow ideas from the natural sciences is not new. The field of evolutionary computation develops stochastic optimisation techniques that are inspired by the processes and principles in evolution. In this thesis, we take inspiration from the evolution of symbiotic associations between organisms. The organisation of ecosystems depends strongly on the interactions between the organisms that share environments (Thompson, 1994). These interactions not only shape the selective forces on organisms, but in addition can be subject to change themselves. Symbiosis, the collaboration between organisms of different types (Mayr, 2001), is very common in nature (Margulis, 1998). Symbiotic associations can alter an organism's biotic environment: the selective context in which it will appear. This modification in environment is most clear where symbiosis is taken to its logical conclusion, symbiogenesis: where the symbionts involved become reproductively inseparable (Khakhina, 1992). Symbiogenesis is thought to have been responsible for several major evolutionary transitions (Maynard Smith and Szathmáry, 1995), including "perhaps the most important and dramatic event in the history of life [. . .] the origin of eukaryotes", (Mayr, 2001, p. 51) when an archaebacterium and a eubacterium in a symbiotic relationship were genetically joined (Margulis et al., 2000). There are also less extreme symbioses between free-living species that still a have significant impact on their likely biotic environments (Thompson, 1994), such as between flowers and their pollinators.

Why might the principles of symbiosis and symbiogenesis be useful algorithmically? There are three concepts that we aim to make use of in problem solving: 1) the incorporation of sub-solutions to other components in the system; 2) a preference for the variant type that is most compatible; and 3) recursion. We illustrate these concepts with the following hypothetical example. Let us suppose that one organism, A, performs some ecological function that is desirable to another type, B. If B can encourage the likelihood that A is present in its environment, B has improved its reliability of access to that function. Moreover, B has not itself had to evolve that function – which may be infeasible from B's genetic position – instead a large, but non-random change is introduced by 'outsourcing' that functionality to A. Let us further suppose that there are two different species that perform the desired function, A and α. It is likely that one of these will be more compatible with B than the other (*e.g.*, a lower overlap in resource usage with B, or by-products are less toxic to B). In this scenario, we might hypothesise that B would preferentially evolve symbioses with α over A.[1] Finally, let us suppose that B encapsulates α such that the pair is reproductively inseparable, resulting in some higher-level function. Now the process can start over, with (B, α) composites in a state ready to evolve symbioses with other types.

This thesis addresses the following questions:

1. How can we automatically identify and exploit structure in a problem to provide scalable optimisation through automatic problem decomposition?
2. What are the fundamental algorithmic properties that enable efficient automatic problem decomposition?
3. What are the corresponding properties of a problem that render it amenable to decomposition?

The original motivating factors in our work are founded in the organising principles in both computational problem solving and evolutionary biology. Within the contexts of problem decomposition and ecosystem structuring, some of the fundamental themes are functional modularity, encapsulation and abstraction. These systems raise challenging questions in both domains. For instance, from an engineering aspect: even in scenarios when some system expertise is available, how appropriate are hand-designed decompositions in comparison to automatically discovered decompositions with respect to reflecting the intrinsic interdependencies? From a biological perspective: what role can the evolution of symbiotic associations play in the structuring of biological communities (*associative evolution*)? In particular, how important are symbiotic associations in the evolution of complexity?

[1]Stating this without intentionality: the variants of B with stronger symbiotic associations with α would be fitter than the B with stronger symbiotic associations A.

1.1 Contributions

In this thesis, we are interested in developing a bottom-up approach to automatically decompose problems in order to match the scale of search to the scale of structure in the problem landscapes, in a recursive manner. We provide a method that is capable of operating without assuming any *a priori* information regarding problem properties, such as the epistatic structure or variable ordering.

The top-level goals of this thesis are:

- **To develop an algorithmic framework that automatically identifies and subsequently exploits the problem structure, thereby providing efficient optimisation.** The algorithmic approach should: 1) encapsulate lower-level components once optimised; 2) respect context sensitivity of modules; and 3) be recursively applicable.
- **To identify the critical characteristics and mechanisms that afford the proposed algorithm efficient optimisation.** Identifying these characteristics should lead to an understanding of the applicability and generality of the approach. The knowledge generated should be in terms of both the problem properties that they can exploit and the substrates in which such an approach can be implemented.

To achieve these goals, this thesis proposes a family of three algorithms. We work through these three variants to develop an understanding of how to best achieve bottom-up decomposition. In particular, an essential feature of our approach is how to automatically identify which lower-level components should be used together in higher-level components: this is the process of forming symbiotic *joins*, and hence enabling *macro-scale* variation. We call the framework that encapsulates all of these algorithms 'MACRO', as the fundamental aim is to create macro-level variation for scalable search. Here we provide a brief description of each of the three algorithms that we present in this thesis.

Hard Joins. A sketch of the first algorithm is as follows:

1. Run multiple hill climbers to find several different local optima;
2. Extract correlations between variables present in these optima;
3. Combine the most strongly correlated variables into macro-variables; and
4. Recurse: repeat from step 1, in the new search space of the macro-variables

When macro-variables are created, they comprise highly coherent combinations of variables, and are not merely large random collections of variables. Therefore, subsequent search can focus on finding combinations of these macro-variables, without the need for further optimisation in the lower levels.

We label this instantiation of our framework MACRO-H to reflect the 'hard' joins that it makes between atomic variables to create macro-variables.

Soft Joins. The joins made in MACRO-H are permanent and irreversible. Thus, the algorithm only joins the most strongly correlated variables. We recognise that in situations where MACRO-H rules out a permanent and complete join, there may still be significant correlations across several local optima. Accordingly, the second algorithm uses the correlation information probabilistically. The overall structure is similar to MACRO-H, although the steps taken by the higher-scale hill climber are created stochastically (rather than deterministically), according to probabilistic biases that are learned from the correlations.

1. Run multiple hill climbers to find several different local optima;
2. Extract correlations between variables present in these optima;
3. Define a set of biases that control future co-variation to reflect the strength of correlations; and
4. Recurse: repeat from step 1, with hill climbers that stochastically create multi-locus variation controlled by biases from step 3.

We label this instantiation of our framework MACRO-S, where the S stands for 'soft', in contrast to the hard joins in MACRO-H.

Serial Visitation, Distributed Join Decisions. The previous two algorithms adapt their biases in correlations according to an ensemble of local optima, which requires a mechanism that can aggregate and analyse information from across such an ensemble of local optima. The third variant algorithm does not depend on the availability of such a mechanism, instead simply using localised feedback to modify biases in future correlations. The algorithm runs a single hill climber to a local optimum at a time, gradually reinforcing co-occurrence at the local optima it visits, rather than inspecting a batch of local optima in parallel.

1. Run a hill climber to a local optimum;
2. Reinforce the bias between variables that are present in that local optimum; and
3. Recurse: use a hill climber that creates multi-locus variation with biases from step 2.

Since the correlations in available variation are introduced gradually, this algorithm still effectively aggregates the information from many local optima. However, the visitation is performed in series. The progressive reinforcement of correlations provides macro-scale variation that resolves modular difficulty efficiently.

The changes made that allow serial visitation also enable the algorithm to be implemented in a distributed manner. Hence, we suffix this algorithm MACRO-D, where D stands for 'distributed'.

The substantive contribution of this thesis is to provide a family of efficient optimisation algorithms that assume no *a priori* information about the system structure. These algorithms offer a bottom-up approach to automatic problem decomposition. The development, validation and investigation of these algorithms leads us to the following thesis:

> The MACRO framework we present can solve, in time sub-quadratic in the problem size, nearly decomposable problems that require exponential time for any local search method. Our method operates by using micro-scale search to guide the formation of structures that enable macro-scale search over several scales of organisation, and we show that this effectively decomposes the problem in a bottom-up manner without using any *a priori* information about the problem structure.

The algorithms comprise two adaptive processes: a rapid exploration, and a slower adaptation of the decomposition that acts on information extracted from the rapid exploration. Changes in the decomposition have the effect of redefining the units of search for the fast mechanism, thereby providing macro-scale search – the units of which were created through micro-scale search. In this manner, *micro-scale search guides macro-scale search*.

1.2 Relationship with Other Evolutionary Optimisation Methods

We discuss the relationship between MACRO and three different evolutionary methods for optimisation: a generic evolutionary algorithm (EA); a memetic algorithm (MA); and an estimation of distribution algorithm (EDA).

An EA has three main stages: evaluating the current population, selecting a fit subset of the current population to reproduce, and applying variation to the selected subset to create the candidates for the next generation (Mitchell, 1996; Bäck, 1996). In MACRO, we also take a generational approach and employ a population (of hill climbers). However, unlike an EA, the specific genotypes held in our population are short-lived: from one generation to the next, new initial conditions are drawn. The variation is also very different: when MACRO adapts the decomposition, it modifies the available variation throughout a run – the variation mechanisms in a traditional EA are not adaptive.

MAs modify the basic EA framework by applying local search to each population member, which determines the fitness used in the selection phase. Crossover and mutation are applied to the selected candidates to modify the inherited genetic material, which changes the seed positions for which subsequent local search is applied (Hart, 1994; Krasnogor and Smith, 2005).[2] Both MACRO and MAs apply hill climbers to members of the current population, and the final states of the hill climbing have an influence on future search. However, the nature of that influence is very different in each case. Memetic algorithms only modify the initial conditions from which local search is applied. MACRO uses information from the final states of the hill climbers to adapt the decomposition, and this creates correlations in the future variation. Thus, future hill climbing is performed at the macro-scale, whereas all local search in MAs is performed at the micro-scale.

EDAs use a population and perform selection in a similar manner to a traditional EA, but differ in the variation stage. They build a statistical model that aims to reflect the dependency structures in the problem by interrogating the correlations between variable configurations in members of the selected subset (Baluja and Caruana, 1995; De Bonet et al., 1997; Pelikan et al., 1999). There is no direct inheritance of genetic material from one generation to the next. Instead, the statistical model is sampled to create variants that take into account how variables depend on one another. MACRO adapts the decomposition that it uses, either by joining variables or by introducing biases into future variation. This decomposition aims to represent the epistatic structure of a problem, sharing the aims of model building in EDAs. However, in an EDA the model is simply used to create new initial conditions that contain elements of the fitter than average candidates from the previous generation. In MACRO, the decomposition defines the future variation, thus leading to macro-scale variation in subsequent search.

The MACRO framework is distinct from all of these approaches. Merely adding local search to a generic EA, or hybridising model building with local search (*e.g.*, Zhang et al., 2004) to repair each of the candidates drawn from the model does not fundamentally approximate our method. Since EAs, MAs and EDAs do not ever modify the units used by the search, even if a multivariate model is constructed that accurately reflects sub-problem structures, the methods do not provide modular variation beyond the initial sampling of that model. Applying micro-scale search to such candidates will not be able to search in combinations of modules.

[2]This is a description of a Baldwinian memetic algorithm. Lamarckian memetic algorithms replace the current population with the locally optimised genotypes, in contrast to only using the local optima to determine fitness.

1.3 Methodological Approach

We aim to understand how to design computational processes that can efficiently provide problem solving. While the general computational efficiency of our resultant processes is important, the understanding gained from taking a principled approach in designing environments in which to test our algorithms and demonstrate principled advantages is central to our development of knowledge.

We define some terms that encapsulate our metrics for success. In this thesis, we consider combinatorial optimisation, and therefore measure the size of a problem in terms of the number of decision variables (Motwani and Raghavan, 1995) – but note that this gives rise to an exponential number of possible combinations in the number of variables. Hence, when we say that an algorithm is *able to solve a problem*, we refer to an ability to find a global optima in time that scales polynomially with the problem size. Conversely, we define an algorithm as *unable to solve a problem* when its performance scales exponentially with the problem size (Weiss, 1997).

We aim to demonstrate in principle scenarios where one process can solve a problem, and other approaches cannot solve the same problem. That is, we are interested in qualitative, and not merely quantitative, distinctions in performance. However, simply showing a distinction in performance without understanding is not wholly satisfactory: we aim to demonstrate algorithmic superiority of proposed techniques, but also *why* these techniques can outperform others. By appropriately selecting test environments that exhibit properties that can be exploited by the algorithm under investigation, and yet without the specific feature that enables such an algorithm to exploit that problem property, other approaches cannot solve that problem, we can gain a deeper understanding of why our proposed algorithm works, and where such an approach will be applicable.

Of course it is important to use test systems that exhibit properties representative of real world problems, and we shall argue why the properties that we have selected are general and relevant.

Scope. We are inspired and motivated by understanding the organising principles influential in both biological and computational systems. Although algorithmically sophisticated and novel, the relatively simple procedures that our proposed framework employs enables us to explore the implications for our understanding of biological organisation when interpreting our MACRO framework as an abstract model of ecosystem evolution.

However, it is important to delineate the form of contributions made by this work. Its major contributions are to the evolutionary computation community: this thesis develops an algorithmic framework for automatic problem decomposition, and provides evidence relating to the computational efficiency of this framework. Analogies with biological processes are provided for interest.

1.4 Overview of the Thesis

This thesis is organised around several contributions. Chapters 3–7 describe technical contributions, while Chapter 8 discusses more general implications of the contributions as a whole.

In Chapter 2 we describe properties and concepts that aid our understanding of systems and their decomposition. Further, we review optimisation techniques that take several different approaches to address problem decomposition, identifying their affordances and limitations.

Chapter 3 introduces a problem generator that exhibits modular interdependency, in a natural and flexible manner. To investigate a variable relationship between algorithms that can and cannot exploit modularity, we develop a problem that has a tuneable strength of modularity present. We call this the variable structural modularity (VSM) problem, and it is used in some later chapters to expose the ability of certain algorithmic approaches in modular domains. Parts of this chapter are published in (Mills and Watson, 2007b).

Chapter 4 briefly describes two studies that lay foundational development in our understanding of the exploitation of modularity. First is a study into the ability of genetic algorithms using linkage-preserving crossover in a modular scenario. We describe conditions sufficient for crossover to provide modular variation, and demonstrate when this is qualitatively distinct from simpler single-scale search processes. Second is the introduction of a compositional algorithm, called the reciprocal synergy symbiosis algorithm (RSSA), which makes some progress in generalising the applicability of compositional techniques over prior methods. This study uses the VSM problem to demonstrate that the decomposition identified by the RSSA corresponds to the problem structure. These two studies have been published in part in (Mills and Watson, 2007b,a).

Chapters 5, 6 and 7 describe the development of the primary algorithmic contribution of this thesis, the MACRO framework.

Chapter 5 builds on the foundational knowledge established in Chapter 4, combining the structural identification from the RSSA with the concept of using multiple scales of search from the genetic algorithm investigation. We introduce MACRO-H, which illustrates the principles behind the coupled multi-scale search processes that comprise the MACRO framework. We illustrate how it can achieve efficient automatic problem decomposition in an idealised test landscape that is computationally difficult for many different forms of search that cannot appropriately evolve a system decomposition. The resulting algorithm is very efficient, and we derive analytical expressions for the expected time to solve both hierarchical and two-layer modular problems. In each case, the proposed algorithm scales log-linearly, despite the cost to single-scale search methods being exponential in the problem size.

Chapter 6 considers MACRO-S, a more general algorithm that uses more flexible association formation rules. We investigate the conditions under which the increased flexibility is advantageous, and consequently, we demonstrate that MACRO-S can solve a broader class of problems that exhibit some ambiguity in local optima. The algorithm in this chapter is presented in (Mills and Watson, 2009).

In Chapters 5 and 6, the instantiations of the MACRO framework both used a batch-based system to evolve an appropriate decomposition. However, the underlying aim of increasing the likelihood of co-occurrence of certain specific combinations of primitives need not require batch analysis. In Chapter 7 we demonstrate that in fact the underlying association formation (problem decomposition) is simple enough to be entirely distributed and implemented in a serial visitation of different local optima. This is exciting for several reasons, including that it demonstrates the simplicity of the concepts needed to provide the type of scalable optimisation that MACRO does. Moreover, this process can be implemented in distributed substrates such as ecosystems.

In Chapter 8 we discuss how MACRO-D can be interpreted as a model of ecosystem evolution – the assumptions that this requires, and the conditions under which our results suggest associative evolution to be qualitatively distinct from a non-associative model. This chapter also brings together some more general concepts that we have provided evidence for in the contribution chapters (3–7), as well as summarising the contributions of the thesis. We also suggest some future directions.

Chapter 2

Foundations

In this chapter, we describe background ideas that assist in understanding the contributions made by this thesis, from two broad areas: computational search and general systems science with particular relevance to modularity.

For a problem decomposition method to gain any traction, the system under consideration must exhibit some structuring that allows a meaningful decomposition, where sub-tasks can be solved in partial isolation. Modularity is one exemplar concept of system organisation that ideally fits with problem decomposition. We shall explain what modularity is, and provide evidence for its generality in real systems. We also cover other related systems-level topics.

We describe a number of search heuristics, including a general overview, and focus on techniques from evolutionary computation that aim to exploit modularity. These include cooperative coevolutionary techniques, estimation of distribution algorithms (EDA), and compositional search. The coevolutionary approaches (CCEA) use many sub-populations that each represent a part of the solution, which is a sentiment that we aim to make use of. However, CCEAs pre-define the roles for each of the sub-populations, which requires *a priori* knowledge that our approach does not need. The compositional search algorithms use explicit encapsulation of lower-level units to scale up the search process recursively. Our approach is influenced by the recursion and encapsulation, but we move towards a more general representation of higher-order search that does not require explicit encapsulation. Moreover, central to our approach is a separation in timescales between creating new variational units and performing search with those units. The similarities with EDAs stem from the use of an external model to represent epistatic dependencies between variables. However, in our approach we use the information stored in the model in a different manner than in EDAs. EDAs sample entire candidates from their model, while we use our model to inform macro-scale variation steps that are used for higher-order hill climbing. Hence, the separation in timescales differentiates our approach from both EDAs and compositional search methods.

2.1 System Science

2.1.1 Definitions of System Properties

Here we provide definitions for some of the more precise terms that we use in this thesis, in particular, with respect to system organisation.

Interactions, Dependencies, Epistasis

All but the most trivial of systems have interactions between at least some of their variables. There are several names given to these interactions: dependencies, epistasis, and linkage.

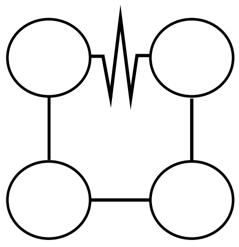
When the state of one variable depends on the states of other variables, we say that there is a *dependency* between those variables. In genetics, the term used is *epistasis*, or *epistatic dependency*: the interaction between multiple genes (Ridley, 2004). Two subclasses of epistatic dependency are polygenic relationships (when several genes all affect a single characteristic) and pleiotropy (when one gene affects several phenotypic characteristics).

Linkage is a related concept that evolutionary computation has borrowed from the evolutionary genetics literature. *Physical linkage* describes the distance between epistatically dependent genes (variables with interdependencies) in the genome (problem representation). When dependent genes are closely organised on the genome, we call this *tight physical linkage*, or a highly correlated epistatic-to-physical linkage map. In general, the organisation of problem variables in a computational problem is not known *a priori*, and thus we can only assume a random epistatic map.

Consistency and Inconsistency

A logical theory that does not contain any contradictions is consistent. Similarly, a physical system where there exists some configuration of sites such that all bonds can be satisfied is *consistent*. More generally, when dependencies are all satisfied, such a system is consistent.

An *inconsistent* system has no possible configuration that satisfies all of its dependencies. Note that this is different from a system in a frustrated configuration, which can occur in a consistent system – this may be at a locally optimal configuration.



(a)

Configuration	Complementary Pair
$+\star-\ \ -\ \ +$	$-\star+\ \ +\ \ -$
$+\ \ +\star-\ \ +$	$-\ \ -\star+\ \ -$
$+\ \ +\ \ +\star+$	$-\ \ -\ \ -\star-$
$+\ \ +\ \ +\ \ -\star$	$-\ \ -\ \ -\ \ +\star$

(b)

Figure 2.1: Lowest energy configurations in an inconsistent four-cycle spin glass sub-configuration, complementary pairs are shown on each row. The $\star$ indicates which bond is not satisfied in this configuration

Frustration

Frustration is a condition that occurs in spin glasses when the configuration of interactions between magnetic sites cannot all be satisfied, but is not identical to an inconsistent configuration. Specifically, when the energy contribution for a particular site is identical for any of its configuration (either if it is an Ising system), frustration occurs (Ramirez, 1994).

Consider the following example Ising spin glass. For a 4-cycle with 3 positive bonds and a single negative bond of equal magnitude, half the states are of optimal energy. There is no way to satisfy all four bonds in this system. Consequently, the configurations with lowest energy are given in Figure 2.1, in which 3 bonds are satisfied and one bond is unsatisfied. Note that the particular bond that is unsatisfied can be located anywhere. The case generalises to simple cycles of any size with a single negative bond.

A single broken bond is sufficient to show the types of structures that must be overcome in frustrated Ising spin glasses. There are additional configurations of bonds that lead to frustration (*i.e.*, with an odd number of negative bonds), which may lead to further computational difficulty.

Degeneracy

When there exists multiple states that have the same energy, a system is said to be *degenerate*. Given our focus on optimisation, we are primarily concerned with degeneracy in local optima. All spin glass systems exhibit some degeneracy, since inverting all spins in a system gives the same energy level. An unfrustrated system is said to be 'singly degenerate'. However, nontrivial multi-fold degeneracy can manifest when frustration is present.

2.1.2 Fitness Landscapes

Wright (1931) introduced the concept of the fitness landscape, which provides a metaphor for how populations may evolve, by organising neighbouring genotypes on a surface and ascribing the fitness of those genotypes to the height of the surface. The concept of the fitness landscape has been the subject of a long running dispute in evolutionary biology, in part because of the counterposition that local optima in low-dimensional representations do not really exist in higher dimensional genotype spaces (see Wade and Goodnight (1998) and Wilkins and Godfrey-Smith (2009) for defences, and Coyne et al. (1997) for counterarguments).

The argument over the presence of local optima for biological populations is not critical to this thesis, as the issue of local optima in computational problems is central to the basic requirement of search techniques. We may use the concept carefully to aid our understanding of why certain mechanisms will be hindered. However it is important to note that, in line with Jones (1995b), we must acknowledge the abilities of different mechanisms for which the local optima in one landscape are not at all problematic for other mechanisms. See in particular Chapter 7 and Mills and Watson (2006) for further discussion of our work on mechanisms that change the effective fitness landscape.

2.1.3 Modularity

The term *module* typically refers to a component of a system, which can meaningfully be considered in partial or full isolation from the rest of the system; the internal dependencies of that component are stronger than its external dependencies. There are several different meanings attached to the term *modularity*, including systems that exhibit functional autonomy (*modular decomposition*), and the repeated use of a modular component (*repeated modularity*). As Schlosser (2002) notes, the latter type requires modular components in order for there to be multiple instances of such a module, indicating that functional modularity is in some sense a more fundamental concept than repeated modularity.

Functional decomposition, the primary class of modularity that we investigate in this thesis, is concerned with systems that have several sub-units whose function is localised and largely independent of other portions of that system. Simon (1969) identifies a system as *nearly decomposable* if the dependence between modules is limited, and those dependencies only influence the module in question in an aggregate manner. (Contrast this with a *separable* system, where there are no dependencies between modules).

Modularity in Natural and Engineered Systems

The pioneering work of Simon (1969) identified how modularity and hierarchy can describe structures that exist in cooperate and social organisation; biological systems as diverse as ecological interactions (Olesen et al., 2007; Martín González et al., 2009), phenotypic development (Wagner and Altenberg, 1996), and genetic regulation (Hartwell et al., 1999; Ihmels et al., 2002; Vilar, 2006; Irons and Monk, 2007) are accurately described as modular. Cognition is often described as broken down into functional modules in the brain (Fodor, 1983). Social systems also frequently exhibit modularity, or 'community structure' (Newman, 2006). Human-designed products are often broken down into subsystems that can be manageable for one designer (Baldwin and Clark, 2000; Huang and Kusiak, 1998), and note that modules are often designed for reusability as well (Smith, 1997).

The Evolution of Modularity

There is a body of work that aims to provide explanations for why modularity is observed in so many evolved systems. This thesis does not directly address the evolution of modularity, or its impact on the evolution of evolvability. However, the overlap of concepts involved in this work with our research are not insignificant. In both lines of work, we are interested in modular organisation keeping interactions 'consolidated' such that optimisation can occur in sub-components without disrupting other sub-components adversely.

Some questions addressed include:

- How does modular organisation affect evolvability? (Wagner and Altenberg, 1996; Kashtan and Alon, 2005; Kashtan et al., 2007);
- Under what conditions is the evolution of modularity encouraged? (Lipson et al., 2002; Parter et al., 2008; Lipson et al., 2001); and
- How does variation in the environment affect the evolution of modularity? (Earl and Deem, 2004; Bogorad and Deem, 1999; Sun and Deem, 2007).

Regularity and Repeated Modularity

The term modularity can be used to describe a system with multiple instantiations of the same component, which is repeated modularity. This concept is sometimes described as regularity, but see Lipson (2007) for a discussion of why regularity does not necessarily invoke modularity. There are several algorithms that aim to exploit this type of structure in a system (*e.g.*, Garibay et al., 2003; de Jong and Thierens, 2004; Walker and Miller, 2005, 2008), and also (Philemotte and Bersini, 2006). These works lie somewhat outside the scope of the research aims of this thesis. We are more interested in addressing the question of identifying modules that reduce the difficulty of the search by reducing the dimensionality (provided an appropriate mechanism), rather than by using a solution discovered in one region in multiple locations in a system.

Several of the test problems used in this thesis (*e.g.*, Chapter 3) are composed from identical modules. However all of these use the same modules as sub-functions as an idealisation to aid understanding, and it need not be the case. Furthermore, the algorithms developed in this thesis provide no mechanism for copying genetic material to different segments of the genome, which would be required in order to exploit a repeated form of modularity.

Defining Modularity

Much work considers what it means conceptually for a system to be modular (Lipson, 2007; de Jong et al., 2004), and how to measure or detect such structure (*e.g.*, Newman and Girvan, 2004; Snel and Huynen, 2004).

Near Decomposability. In a strictly independent module, variables depend only on other variables within that module. While this degree of separability may be appropriate for some scenarios, one may more generally question why multiple entirely independent components are considered as part of the same system. Simon's concept of *near-decomposability* describes a more realistic situation (Simon, 1969). Crudely, interactions between modules are weaker than those within modules, but they are not negligible. He defines two conditions which describe near-decomposability:

1. The short-run behaviour of each module is approximately independent of the short-run behaviour of the other modules; and
2. The long run behaviour of modules depends only in an aggregate way on the behaviour of the other modules.

—Simon (1969, p198).

Modular Interdependency. Where Simon's definition of near decomposability focuses on dynamical systems, Watson and Pollack (2005) refine this definition to one that is more suitable for computational search problems. A module can take a number of different settings, C, according to its size. For a binary representation of k bits, $C = 2^k$. Watson and Pollack define C' as the number of settings for that module that are optimal for some context, and this can take values in the range $[1, C]$. If C'=1, the system is separable. Problems that are separable are often trivial, and there is no room for higher-order interactions (*e.g.*, hierarchically organised dependencies). If C'=C, the system is not decomposable and no modularity is present. If $C' < C$, the system is decomposable. Problems that are not decomposable are of little interest within a study on the exploitation of modularity. Problems that are decomposable but not separable ($1 < C' < C$) are of the most theoretical interest. Watson and Pollack define systems for which $1 < C' < C$ as exhibiting *modular interdependency*.

2.2 Heuristic Search

Stochastic search algorithms are often employed in situations where a complete method (which can guarantee optimality) is infeasible. That is, time exponential in the size of the problem is required to solve that problem (even if an approach is smarter than enumeration, application becomes computationally infeasible very rapidly with even modest problem sizes). Blum and Roli (2003) provide a survey of search heuristics, and distinguish between trajectory methods and population-based methods. Trajectory methods maintain a single point in the search space at any one time. These include gradient ascent, random mutation hill climbing, and simulated annealing. Population-based methods aim to maintain a diverse set of candidate solutions, in an attempt to avoid discovering only local optima. The genetic algorithm is a canonical example of a population-based heuristic, and is a member of the class of evolutionary algorithms that also use principles of 'survival of the fittest' from evolutionary theory (Bäck, 1996).

Gradient ascent assumes that the form of a fitness function is known, and is differentiable (perhaps twice, depending on the method), such that from any position, a new position can be calculated according to the shape of the fitness landscape (Luke, 2009). Many stochastic search methods use the weaker operating assumptions of black box optimisation (BBO, Goldberg, 1989; Droste et al., 2006). These assumptions stipulate that the only access to information about the function is via evaluation of a fully specified candidate solution. When gradient information is unknown, we can test all neighbouring configurations and accept that which improves the solution quality the most (steepest ascent hill climbing, Forrest and Mitchell, 1993a). A cheaper alternative local applies random perturbations to the current configuration, accepting the change if it improves solution quality (random mutation hill climbing, Forrest and Mitchell, 1993a). Simulated annealing follows a similar process of applying random perturbations, and always accepts changes that improve solution quality. In addition, changes that reduce quality can be accepted probabilistically, according to the system 'temperature', which reduces over time ('anneals') such that early on most changes are accepted regardless of the difference in quality, and towards the end of a run, only changes that make improvements are accepted (Kirkpatrick et al., 1983).

Evolutionary algorithms have received much attention in the field of computational intelligence and heuristic optimisation, and accordingly there are many variants. The genetic algorithm (GA, Holland, 1975; Goldberg, 1989; Mitchell, 1996) maintains a population of candidate configurations, selects a high fitness subset for reproduction to the next generation, creating offspring by applying variation to the selected parents with mutation and crossover operators. An evolutionary strategy implements a similar protocol but typically only applies variation through mutation. In $(\mu + \lambda)$ strategies, the current population (μ) competes with the mutants (λ) for space in the next generation (Beyer and Schwefel, 2002). In (μ, λ) strategies only the mutants compete for space in the next generation, thus a slightly closer resemblance to genetic algorithms. Other related work includes that on evolutionary programming, which searches for programs rather than fixed length, discrete variable search spaces (Koza, 1992; Looks, 2006); and ant colony optimisation, which reinforce successful (and short) paths to resources via 'pheromone trails' laid by artificial ants (Dorigo et al., 1996).

The Building Block Hypothesis

The building-block hypothesis (BBH) was developed as an explanation for the success of the genetic algorithm. The BBH involves the concept of discovering building blocks (crudely: modules, although see below for the important differences) and subsequently mixing those building blocks to find higher-order solutions.

A schema specifies values for a subset of variables in a binary string; this defines the set of all strings which contain those values. A building block is defined by Goldberg (1989) as a fit, low order, short schema. Tight physical linkage is preserved by crossover, so elements within short schema are likely to remain neighbours. By maintaining partial solutions in the form of building blocks, and combining multiple building blocks by crossover, we should find fitter higher order schema. This should result in the discovery of high fitness solutions since each parent can contribute solutions to different building blocks through crossover. This is the essence of the building block hypothesis.

The BBH offers an attractive way of thinking about how crossover should be able to exploit modularity. However, it has not been straightforward to utilise this hypothesis. Identifying problems that exhibit modular properties of the right kind to allow solutions to be more easily accessible to algorithms which exploit these modular properties has been unsuccessful in several attempts (*e.g.*, Forrest and Mitchell, 1993b; Spears, 1993).

This thesis does not directly address the BBH. Instead, we take a parallel approach that shares some of the intuition of the BBH, while remaining agnostic about the abilities of standard genetic algorithms. In particular, we aim to a) identify sub-solutions; b) search in the space of sub-solutions to find combinations that create higher-order sub-solutions; and c) use the ideas of recursion implicit in such an explanation. Where our approach diverges from the BBH is illustrated by the definition of building blocks with respect to modules. In our work,

> a) *modules need not have short defining length*: we are interested in solutions that do not depend on prior knowledge of the physical to epistatic linkage map (*i.e.*, no tight linkage assumption);
> b) *modules need not be small*: their solutions need not be discoverable by enumeration (which is the case when the size of building blocks is bounded by a constant).

Throughout this thesis we shall consider problems that exhibit modularity according to these criteria to test our proposed algorithms.

In this thesis, the most closely related work comes from the field of evolutionary computation, and in particular the class of estimation of distribution algorithms (EDA). We review the concept of EDAs and work in the area thoroughly below. In addition, there are several computational methods that aim to provide automatic problem decomposition, and within this category, some that are explicitly compositional approaches. We describe each of these types of algorithm and work in these areas in the following sections.

2.3 Estimation of Distribution Algorithms

Estimation of distribution algorithms and GAs have common features, such as the use of a population of candidate solutions, and stochastic selection based on fitness of these solutions to determine which areas of the search space will be explored further. However the process of inheritance of genetic material from one generation to the next is very different. In an EDA, the selected subset of the population is used to build a probabilistic model, aiming to identify what made that subset 'fit'. It is this model that is sampled to produce new candidate solutions, rather than the mutation and crossover operators used in GAs.

A key motivating factor in using probabilistic models to explore the problem space is that it opens up the possibility of using mature statistical and machine learning techniques in conjunction with the advantages of population based search. A second very important factor in this approach is an attempt to identify the epistatic dependencies in problem structures, sometimes known as *linkage learning* (Harik, 1997).

The algorithmic framework that we present in this thesis evolves correlations between species to provide specific multi-locus variation that reflects the macro-scale structure in a problem. The evolution of these correlations are also a form of dependency identification, a process that can be applied with no *a priori* assumptions of how the problem is structured. An estimation of distribution algorithm aims to extract dependencies in a problem by modelling what was good, or common, to an above-average-fitness subset of candidates. The model is then sampled to produce new candidate solutions that take account of the dependencies that have been discovered. Building a model of the epistatic dependencies between problem variables then, is integral to both of these approaches. Accordingly, we use this section to describe a variety of approaches to model building.

The idea of building a model to estimate underlying features of a population, and using that model to influence the regions of the search space that will be explored is broad and encompasses many approaches. In this section we consider key regions from this diverse set of techniques. For more exhaustive detail on estimation of distribution algorithms, we refer the reader to these reviews on EDAs (Pelikan et al., 2002; Larrañaga et al., 1999) and of close relatedness, on linkage learning (Chen et al., 2007), as well as two recent books that describe algorithm developments and applications (Lozano et al., 2006; Pelikan et al., 2006b).

One of the key defining characteristics in an EDA is the model, and specifically, the class of interactions that it can represent. We follow (Pelikan et al., 2002) in using the model richness to organise our descriptions of these algorithms, separating into univariate, bivariate and multivariate categories. In addition, we identify a further category: EDAs that use models with undirected dependencies.

2.3.1 Univariate EDAs

This class of algorithm considers the distribution of alleles at each variable independently from the other variables, hence the name 'univariate'. While our primary interest in EDAs is their ability to model interdependencies between variables, these univariate algorithms initiated research in the field of EDAs and illustrate some of the other concepts important to EDA design.

Population Based Incremental Learning

Population based incremental learning (Baluja, 1994; Baluja and Caruana, 1995) is a simple algorithm that represents the state of the population using a probability vector, which describes a distribution of variable states. A population of samples is drawn from the distribution defined by that vector, and the fittest individuals are selected to update the probability vector. The vector assumes variables to be independent. Each variable in the vector is moved towards the univariate frequency observed for each variable in the selected subset, the distance controlled by a learning rate. The pseudocode for PBIL is shown in Algorithm 1.

The original algorithm described uses a single solution to update the model (*i.e.*, E=1), but it can be applied to larger sets in a straightforward way. The authors also suggest learning from both positive and negative evidence, by moving towards the top E solutions, and away from the worst E solutions (Baluja and Caruana, 1995). A further extension is the parallelisation and incorporation of a crossover operator (Baluja, 1997).

PBIL has been shown to outperform a GA on several problems, including some that were designed to be 'GA-friendly'. The authors attribute the success in part to the cost of parallel evaluations that are not as effective when the population of a GA has converged. However, since each variable in the problem is considered independently the model cannot represent any interdependencies between problem variables.

Algorithm 1 Population-Based Incremental Learning

```
Define:
    for a problem with $n$ binary variables
    $\lambda$ is the learning rate
    $P$ is a population of solutions
    $\mathbf{w}$ is probability vector, $0 \leqslant w_i \leqslant 1$
Initialise:
    $\mathbf{w} \leftarrow w_i = 0.5, \forall i, i \in [0, n-1]$
Main loop:
while !(reached iteration limit OR $\mathbf{w}$ converged) do
  $P \leftarrow$ draw $|P|$ solutions from distribution $\mathbf{w}$
  $P_s \leftarrow$ sort the solutions in order of fitness
  $S \leftarrow$ select the fittest $E$ solutions from the population, and use these to update the model:
    for $i \in [0, n-1]$, calculate the frequency, $f_i$, of candidates in the top $E$ where $x_i = 1$
  move the probability vector towards this frequency:
    $w_{i,t+1} = w_{i,t} + \lambda \cdot f_{i,t}$
end while
```

Bit-Based Simulated Crossover

Whilst not fitting so neatly into the EDA framework described earlier, bit-based simulated crossover (Syswerda, 1993) was arguably the earliest of all evolutionary algorithms to consider distributions of allele values. This abstraction was motivated by the understanding that uniform crossover randomly assigns the offspring of an allele at any locus where the two parents have disagreeing values, and not changing the sites where parents have the same allele. Instead of biasing a particular offspring to the genetic material from two particular parents, the probabilities of each variable are calculated over the population, and these probabilities are used to bias the allele assigned to offspring.

The Univariate Marginal Distribution Algorithm

The Univariate Marginal Distribution Algorithm (UMDA; Mühlenbein and Paaß, 1996; Mühlenbein, 1997) is another univariate EDA. It is similar to PBIL, although the description focuses more on maintaining a population (note that the inheritance is equivalent: only through the marginal distribution and no direct copying of solutions). The model is constructed from the univariate marginal probabilities $p(\mathbf{x})$ of the selected E high fitness individuals. Note that this does not use an incremental update to the probabilistic model, but instead constructs a new model from the current high fitness individuals in each generation.

The Compact Genetic Algorithm

The compact genetic algorithm (cGA; Harik et al., 1999) is another method that considers variables independently. Like PBIL, it uses a probability vector to represent the univariate probabilities. This distribution is used to generate two samples, and these compete. Wherever the alleles differ, the probability vector is updated towards the fitter solution, with an update rate of λ (described as set to $1/n$). Where the alleles agree, no change is made to the probability vector.

2.3.2 Bivariate EDAs

When the fitness contribution of a particular variable X_i (marginal fitness) can be calculated by the sum of several terms, each of which has a maximum of two variables (X_i and X_j), we call this a problem constructed from pairwise interactions. One simple example is the Ising model on a lattice, where bonds can be considered as pairwise interactions. A simple example of a function that has higher order is a k-bit trap: each fitness value depends on the unitation of the entire subfunction, which requires the knowledge of states of k variables.

BMDA

The Bivariate Marginal Distribution Algorithm (BMDA). The main procedure of BMDA is very similar to that of PBIL: a model is built from a high-fitness subset of the population, and the new generation is drawn from the model distribution. However this algorithm makes some advances over PBIL since it can consider order-2 interactions (Pelikan and Mühlenbein, 1999). The model is constructed using Pearson's χ^2 statistics to give a measure of pairwise dependencies. A dependency graph containing all of the problem variables is produced; the resulting graph is not necessarily connected (*i.e.*, there may be more than one component in the graph). Since it is acyclic, a straightforward procedure exists to generate new individuals.

MIMIC

The mutual information maximising input clustering (MIMIC; De Bonet et al., 1997) algorithm uses a chain of dependencies to represent the problem structure. It uses a greedy algorithm to construct an ordering, which aims to describe the linkage order in the problem. A probabilistic model is constructed from this chain, which is sampled to construct the new population.

Decision Trees

Baluja and Davies (1997b) present an extension to MIMIC, modelling the dependencies in the problem with a decision tree instead of a chain of dependencies. Its model building finds the minimum spanning tree based on the bivariate frequencies of variables in the selected subset of the population. The tree connects all variables, and thus all variables have a parent except the root. Unlike the model in MIMIC, multiple variables can share the same parent. Sampling this distribution to create the new population is straightforward.

COMIT

Baluja and Davies use the decision tree EDA as the core of 'combining optimisers with mutual information trees' (COMIT; Baluja and Davies, 1997a). Instead of sampling the probabilistic model to directly create offspring for the next generation, COMIT uses the samples to select a seed point for a simpler optimisation algorithm (in this case, a restart mutation hill climber; in (Baluja and Davies, 1998) they use PBIL). Several candidates are sampled, evaluated, and the single fittest is used as a seed point for the hill climber. The local optimum found by this hill climber is inserted into the population. See Algorithm 2.[1]

Algorithm 2 Pseudo-code for COMIT

```
Initialise:
    $P \leftarrow$ random strings of length $N$
while !(termination criteria) do
    build mutual information tree model, $M$, of $P$
    sample model to produce $O$, where $|O| = K$
    evaluate all offspring in $O$, and select the fittest string, $H$.
    hill climb from initial point $H$, to give $H'$
    replace some of P with copies of $H'$
end while
```

STAGE

This algorithm shares some of the motivations of COMIT, and while it does not fit neatly into the category of estimation-of-distribution algorithms, its similarities merit discussion at this point. STAGE comes from machine learning, and is a multi-restart method to select promising initial configurations for a local search method (Boyan and Moore, 2000; Boyan, 1998). Their approach aims to learn the objective function using the trajectories of a simple local search algorithm as training data. Specifically, for

[1] In (Baluja and Davies, 1998) the authors suggest the use of Bayesian networks as a richer model but do not proceed with this enquiry on account of finding optimal networks being NP-complete (although they do identify Heckermann's heuristics).

each starting configuration x_0, the local search algorithm provides some locally optimal configuration, x^*. STAGE constructs a function approximation of the likely fitness of output configurations based on initial configuration, using linear or quadratic regression. Hill climbing is then performed on the resulting function (*i.e.*, in an auxiliary space, not the original problem search space), to bias the next initial condition for the local search, hopefully leading to a high quality solution. These two steps are repeated until a global optima is found. Pseudo-code is given in Algorithm 3.

Algorithm 3 Pseudo-code for STAGE

```
Initialise:
    $x_0 \leftarrow$ random initial state of length $N$
while !(termination criteria) do
  (A) $\mathbf{x} \leftarrow$ trajectory of local search algorithm $\pi$ from $x_0$
  $y \leftarrow x^*$, i.e., record the maximum fitness point on that trajectory $\mathbf{x}$
  add $\{F(x_i) \mapsto y_i\}$ to training set for each $x_i \in \mathbf{x}$
  (B) $\tilde{V}^\pi \leftarrow$ trained evaluation function from training set
  (C) $\mathbf{z} \leftarrow$ trajectory of stochastic hill climbing on $\tilde{V}^\pi$
  (D) $x_o \leftarrow z^*$, i.e., set initial point for $\pi$
end while
```

DSMGA

The dependency structure matrix driven genetic algorithm (DSMGA) was proposed by Yu et al. (2003). It is an interesting hybrid that sits between GAs and traditional EDAs. Variation is applied directly to the population rather than via drawing new samples from a model, so inheritance is direct; and yet that variation is controlled by a model of the problem structure. It constructs a dependency structure matrix (DSM) that describes an estimate of where dependencies exist in the problem, based on the entire population. The second stage is to find a locally optimal clustering of this data, to give an indication of the structure present in the problem. The third stage is to use the information presented by this clustering as a definition for where building block structure exists in the problem, and perform 'BB-wise crossover' on selected individuals. In short, this third stage acts similarly to uniform crossover, except crossover points are excluded within building blocks (see also Sastry and Goldberg, 2004).

The DSMGA has similarities with both GAs and EDAs. The algorithm constructs a model of dependencies in a problem to address unknown linkage, and that model is built in every generation using direct statistical analysis of the population. Inheritance of genetic material is as per a GA: variation is applied directly to parents in G_n to create offspring in G_{n+1}. However what makes the DSMGA stand out from other EDAs and GAs is that the resulting DSM reveals the structure of the problem, in a manner that appears intuitive to the human eye. This may have some value in cases where real-world problems cannot be solved without skilled knowledge. Contrast this with the final model from a successful run of, for example, BOA. Here, because the population has converged at the end of a run, the model need only describe univariate frequencies and will not represent any dependencies between variables.

A hierarchical version (Yu and Goldberg, 2006) is described in section 2.3.3.

2.3.3 Multivariate EDAs

Problems that have complex interactions may require more sophisticated models to accurately represent the structures. Some EDAs have been developed that use graphical models such as Bayesian networks, which are capable of representing a richer class of interactions than the bivariate algorithms reviewed above. A second approach to handling more complex interactions is by recursively applying a simpler model. We describe some algorithms of each category.

The Extended Compact Genetic Algorithm

The Extended Compact Genetic Algorithm, or ECGA, identifies clusters of variables in the problem structure using a greedy algorithm and a minimum description length metric (Harik, 1999). It inherits little other than name from the compact GA (see 2.3.1), requiring a substantially sized population to build a model from, and the model itself is significantly more complicated.

The model is built on a selected subset of the population. The greedy algorithm that constructs this model initially assumes N subsets (*i.e.*, one for each problem variable), and attempts to group those subsets into larger subsets. If the overall 'marginal product model' is improved the grouping is retained, and this process is repeated until no more groupings are accepted. The frequencies in the model are sampled to create an entirely new population in each generation. The model quality is judged on two criteria: (1) the storage required to represent the problem partitioning, and (2) the storage required to represent any nonzero frequencies of pattern occurrence within the population for each subset in the model.

Harik demonstrates the model to discover a decomposition that reflects the building blocks in a concatenated trap (of fixed size) function.

Hybrid Extended Compact Genetic Algorithm

Lima et al. (2005) introduce the hybrid extended compact genetic algorithm, by combining the ECGA with BB-wise mutation and BB-wise crossover (the DSMGA uses BB-wise crossover). The ECGA model building method is used to decompose (partition) the space according to a selected subset of the population, and the two BB-wise operators are used to provide variation. BB-wise mutation involves searching all 2^k configurations in a module, and retaining the single configuration with the highest fitness.

This is an interesting idea, but we find three critical problems with its applicability: 1) the cost of search in module space is exponential (unless k is fixed with respect to problem size, N); 2) applying these operators depends on having found a useful decomposition, which takes a large population size; and 3) there is no concept of handling the possibility that a module might have multiple optima, the best of which is dependent on the context.

The Bayesian Optimisation Algorithm

The Bayesian optimisation algorithm (BOA) utilises a Bayesian network that is capable of representing multivariate dependencies (Pelikan et al., 1999). The probabilistic model is built on the fly from a subset of candidates of above average fitness from the population, and updated in each generation. Bayesian network structures are explored for one that best fits the data from the selected set, using a greedy heuristic. In addition to the dependency graph, each vertex requires a table describing the conditional probabilities of each variable value given the configuration of its parent vertices. These two components make up a candidate Bayesian network, several of which are constructed during this greedy search. Valid dependency graphs include fully connected trees as well as a set of disconnected components.

Generating new samples is similar to the method used by the bivariate EDAs described above. Since the dependency graph is acyclic, it will always contain some roots (vertices with no parents). These roots can be assigned values according to the corresponding univariate probability in the model. These new samples replace a portion of the population, and the next generation begins with selection again.

The additional complexity available in the model, compared to a bivariate dependency model, has benefits and drawbacks. The model is able to represent structures that allow the algorithm to efficiently solve building block problems that other algorithms cannot, such as 6-bit bipolar traps, when algorithm parameters are appropriately set (Pelikan et al., 1999). However, the larger the order of dependencies in the problem, the more complicated the Bayesian network must be. The search for a good network is straightforward if e, the maximum number of incoming edges in the graph, is 0 or 1. But for the general case where $e > 1$, finding an optimal graph is an NP-hard problem (in e). Therefore e is kept low, and a greedy heuristic is used, which adds a random edge, and retains the edge if the network score is improved. The quality of a graph in this search is measured using the K2 variant of the BD metric as proposed by Heckerman et al. (1995).

A variant, hierarchical-BOA, was developed to address dependencies greater than order-k in a hierarchical fashion (Pelikan and Goldberg, 2001a). hBOA is described further below. Another variant was proposed to eliminate some of the cost of model building complexity of the Bayesian network (Pelikan et al., 2008). This makes small increments to both the structure and marginal parameters from generation to generation, and eliminates the requirement for a population (inspired by the compact GA, but see also the EBNA in Section 2.3.3).

Estimation of Bayesian Network Algorithm

The estimation of Bayesian network algorithm follows the classical EDA structure, using a Bayesian network to model dependencies (Etxeberria and Larrañaga, 1999). EBNA has much in common with BOA. However, there are implementational differences, including: a) the Bayesian network is calculated incrementally from the network of the previous generation; and b) scoring the Bayesian network uses a different metric (the Bayesian information criterion). Larrañaga et al. (2000) investigates different Bayesian network metrics in the EBNA, finding that both Bayesian information criterion and the K2 metrics (as used in BOA) are successful.

FDA and LFDA

The Factorised Distribution Algorithm (FDA) uses a model that is supplied a priori for a given problem (Mühlenbein et al., 1999). When the supplied distribution model is accurate, the algorithm is efficient. However, the dependence on the prior specification of the model limits the applicability of the FDA.

The learning factorised distribution algorithm (LFDA) extends the FDA so that it learns the structure using a Bayesian network (Mühlenbein and Mahnig, 1999). The authors acknowledge the approach of learning the network from fitter individuals taken by BOA (see 2.3.3), and choose an alternative model scoring metric, but the algorithms are largely similar.

DSMGA+

This is a recursive version of the DSMGA (see Section 2.3.2), inspired by the success of hBOA (Yu and Goldberg, 2006; Yu, 2007).

The main difference from DSMGA to DSMGA+ is the inclusion of the ability to recurse, via what the authors call ‘substructural chromosome compression’. As per the earlier algorithm, pairwise dependencies in the problem are estimated from the population and used to construct a DSM. Clusters are detected in this DSM via a search heuristic to provide an estimate of the problem structure, which are used for two main purposes. Firstly, when a building block converges (*i.e.*, there are only a handful of different schema present in the population for the variables in that block), the algorithm attempts to compress the block into a single binary variable. This is performed when the difference between the second and third most frequent schemata becomes significant enough. As implemented, this occurs when the difference in normalised frequency exceeds five standard deviations. A second criterion based on mutual information is also applied before a block can be compressed. Once compression has been applied, it is not reversible. The mapping is as follows: the most frequent schema is mapped to 1, the second most frequent schema is mapped to 0, and when any other schema appears in the population it is mapped randomly to 0 or 1.

This second criterion is applied in an attempt to alleviate problems from compressing higher level blocks before lower level blocks are fully converged, and may be restricted to helping the algorithm work correctly in problems where the building blocks are arranged in neat layers. However since it is only tested on problems in this class there is no evidence one way or the other (A random problem from the hierarchical problem generator (de Jong et al., 2005a) may well be problematic). A further concern with the compression mechanism employed is that a building block can only be compressed into exactly two configurations, which appears to be too rigid. This is not manifested in the tests performed since all blocks have exactly two (at least locally) optimal configurations, but it is not hard to imagine that some problems will have multiple optima in one block and only a single in others.

The DSMGA+ is tested on HIFF, HXOR and hierarchical Trap functions and achieves approximately $\mathcal{O}\big(N^2 \log N\big)$ performance on each problem type. The authors also emphasise that the DSMGA+ has an ability to present the structure of the problem it has solved, although it is not quite as neat as when working on flat separable problems.

hBOA

The hierarchical Bayesian optimisation algorithm (hBOA, Pelikan and Goldberg, 2001a,b) is a development on BOA (see Section 2.3.3). It aims to improve efficiency of its representation of dependencies in the probabilistic model it builds in order to solve problems with high-order dependencies where BOA would require exponential time in the problem size. The major change is the use of local structures in the Bayesian networks. In particular, Pelikan and Goldberg (2001a) define hBOA with decision graphs as the local structure, although results are provided using the simpler decision trees (in (Pelikan and Goldberg, 2001b) and later publications, *e.g.*, Pelikan and Hartmann (2007)). Additionally, a diversity maintenance mechanism is used to determine which individuals in the population a newly generated candidate should replace (called restricted tournament replacement, after (Harik, 1994); very similar to deterministic crowding (Mahfoud, 1992)).

hBOA can efficiently solve hierarchical traps (Pelikan and Goldberg, 2000) and HIFF (Watson et al., 1998): curve fitting on performance on the hierarchical traps as reported in (Pelikan et al., 2003) indicates $\mathcal{O}\left(N^{1.63}\log N\right)$.

2.3.4 Models with Undirected Dependency Structures

As discussed in Section 2.3.3, some of the most powerful EDAs use a Bayesian network to model dependencies while others use trees or chains, which are all types of directed graph. However, if the dependencies between variables are not directed, a directed graph may not be able to model the structure accurately. This has motivated some researchers to explore EDAs that use undirected graphs at their core.

The Markov Network Factorised distribution algorithm uses a Markov network to represent undirected dependencies between sets of variables, and identifies (maximal) cliques in this graph to construct a junction graph (Santana, 2003). This junction graph is the final model that is sampled to generate a new population. This framework was later modified to use Kikuchi approximations (Santana, 2005; Santana et al., 2006).

Shakya et al. (2004, 2006) introduce the 'distribution estimation using Markov random fields' framework, which uses Markov fields as the probabilistic model (DEUM). Several instantiations exist, including the univariate DEUM$_{pv}$ (Shakya et al., 2004), and a bivariate version Is-DEUM$_g$ (Shakya et al., 2006; Shakya, 2006).

DEUM$_{pv}$ follows the framework of a typical EDA, and performs a singular value decomposition to find its Markov model parameters, which are used to inform the direction of update to the univariate probability vector (similar to PBIL). The probability vector is sampled to create new candidate solutions as in PBIL. Is-DEUM$_g$ is applied to the $\pm J$ Ising model on a 2D lattice, and uses the lattice structure *a priori*, therefore not addressing the question of where the dependencies are in a problem. Creating a new candidate solution requires a 'Gibbs sampler' to be run many times on the Markov model. This does not use fitness function evaluations, but the authors note that computation time is dominated by the sampling (Shakya et al., 2006). This makes it hard to directly compare performance with more traditional evolutionary algorithms.

Another algorithm in this category is the estimation of dependency networks algorithm (Gámez et al., 2007). This identifies relationships according to high mutual information and low degree of relation, and constructs a dependency network (which is similar to a Bayesian network, but allows bi-directional edges). New candidates are drawn from the model using a Gibbs sampler.

This is an interesting direction for EDAs, presenting the potential advantage of representing dependency structures between problem variables more closely with undirected models. However, a large limitation is that non-trivial methods must be used to sample the distribution. In any directed graph, there will always exist some root nodes whose values can be set according to univariate frequencies, thus providing the values necessary to sample variables lower in the tree. No such guarantee exists for undirected graphs, and the penalty is manifested in the cost of Gibbs sampling or similar.

2.3.5 Properties of EDAs

In the previous sections we covered many different algorithms that fit under the category of estimation of distribution algorithms. The fundamental property that these algorithms share (with few exceptions) is that new candidate solutions are drawn from an auxiliary model. However, other properties characterise the differences in these approaches, and we identify a number of these here.

First, we identify two properties that are important in how effective the resultant algorithm is: 1) the auxiliary model capacity; and 2) the method by which the model is sampled to generate new candidate solutions.

Second, there are several properties that do vary in the algorithms described, but either the alternatives reduce to one another, or they are the type of property that need some decision, but many choices are suitable. These include: 1) whether the model is updated incrementally, or regenerated after each round of selection; 2) whether any of the population progresses to the next generation, or the entire population is drawn from the model in each generation; and 3) certain characteristics of the model construction algorithm.

We describe the properties in each of these two categories. The model capacity is central in defining the capabilities of an EDA – and indeed, following Pelikan et al. (2002), we chose to organise the above descriptions by this feature. The univariate algorithms potentially provide a less biased representation than an explicit population. Once a model can represent dependencies (as in the bivariate or multivariate EDAs), the structure used by that model can influence how well the dependency structure can be estimated. For instance, the use of chains in MIMIC is less general than the forest of tree structures that BMDA allows. Furthermore, the model can either use continuous dependencies (*e.g.*, PBIL), or define a statistical threshold for inclusion of a discrete dependency (*e.g.*, the ECGA).

Sampling the model is key to the algorithm's ability to generate variants in order to search a problem space. Although this contributes more to the algorithm cost than the underlying principle of the procedure, it is nevertheless a central consideration in developing efficient processes. When the model uses an acyclic graph (such as MIMIC, BMDA, FDA, and hBOA; as well as PBIL and UMDA in a degenerate sense), probabilistic logic sampling (Henrion, 1986) can generate a sample in polynomial time. Conversely, the undirected graphs used by the Markov network EDA require a more involved and costly sampling procedure.

The model may be updated incrementally, or regenerated after each round of selection. However, simply using a large population and weak selection will introduce more 'inertia' into the model even if it is regenerated in every generation, mimicking an incremental update. Similarly, the proportion of new candidates introduced to the population in each generation also influences how rapidly the model is updated. These two factors are important in avoiding the introduction of too strong a bias from modelling arbitrary correlations, but are set by parameters of the model rather than the structure of the algorithm, *per se*. One final aspect is the algorithm used to construct the model. Some EDAs use models that require a search in the auxiliary space (*e.g.*, MIMIC, BOA, EBNA) whereas others can be calculated directly from the statistical measures of the selected subset (*e.g.*, PBIL). The former category may use a more powerful statistical model, but in some cases this may be partially hindered by the cost of search in the model space. Baluja's decision tree model uses a minimum spanning tree, which is a polynomial search. However, finding the best-fitting Bayesian networks is an NP-complete problem, and normally a heuristic is used to complete in polynomial time (see *e.g.*, Heckerman et al., 1995).

On the Quality of Models

As mentioned above, there are several different approaches to the model building stage in EDAs. Model choice, and how high quality the model should be with respect to the sampled data, are open questions (see *e.g.*, Hauschild et al., 2007; Echegoyen et al., 2007). Santana et al. (2005) looks at one aspect of model quality in an investigation into 'benign' and 'malign' interdependencies. If, for a pair of variables, the most probable bivariate configuration is predicted by the univariate probabilities, this is labelled as 'benign', and the authors suggest that this type of dependency might not add anything to the model. (Conversely, malign dependencies occur in the case where the bivariate probability has a different result from the univariate probabilities). Experiments with variants of EDAs that model either all dependencies, or only malign bivariate dependencies provide support for this position.

Note that although BOA uses a greedy heuristic to construct suitable Bayesian networks, if a higher quality Bayesian network is found to be necessary, EAs have been used to address such a problem (Larrañaga et al., 1996; Myers et al., 1999). These methods are compatible for use within the Bayesian network construction stage in BOA, EBNA or LFDA.

2.4 Automatic Problem Decomposition

2.4.1 The Principles of Problem Decomposition

Breaking down a problem into sub-components may allow us to solve a problem more efficiently, if the system has a feasible decomposition that facilitates this. Take the example of Tempus and Hora (Simon, 2002), the watchmakers. Each builds watches of 1000 components. Tempus' strategy is to put together sub-components of 10 pieces, then assemble these into sub-components of 100, and finally assemble the overall watch with ten of these 100-piece units. Hora on the other hand simply assembles full watches with no intermediate assemblies. While there is no search problem in this story, the benefit of Tempus' strategy is seen when each watchmaker is interrupted. When Hora is interrupted, he puts down whatever partial watch he was working on, and it dissolves – he has to start over after any interruption. Tempus might lose the progress on the current assembly (be it a 10-part or 100-part unit, or an entire watch), but the dissolution only goes back to the unit at the level below. Thus, interruptions are very costly for Hora (on average, the time taken to assemble 500 components), but Tempus only loses the time taken to assemble 5 components.

Breaking down a task into sub-tasks can be much more efficient – in binary combinatorial search spaces, solving an N bit problem in one monolithic attempt has a set of 2^N possible configurations. Dividing this in two, solving independently, and reassembling has a far smaller search space of $2^{N/2} \cdot 2$. The smaller the components, the greater the reduction on the search space: consider a completely trivial problem that could be broken down such that each variable was an independent sub-task. Here, the decomposed route has a search space of $2 \cdot N$, where enumeration is only linear in the problem size; whereas enumeration in the original space is exponential in N.

However, most problems are not so trivial that every variable is independent of every other variable. Once there are significant epistatic interactions, the main assumption made when applying automatic problem decomposition is that there are meaningful decompositions that can make qualitative reductions in problem complexity, *i.e.*, that sub-components of manageable size exist.

2.4.2 Prior Methods

Given the obvious potential benefits from problem decomposition, many different approaches have been suggested in the literature. Most problem decomposition in computation is top-down: the system decomposition is known beforehand, or defined by the programmer (Grama et al., 2003). In evolutionary computation, we are interested in bottom-up solutions, where the decomposition is not known *a priori*. Instead, discovering the system decomposition *in order* to find optima.

However, many of the attempts to provide automatic problem decomposition have also used some level of knowledge with respect to the system structure. Using such *a priori* knowledge restricts their applicability to problems where such labelling is available.

2.4.2.1 Coarse Graining The Search Space

Kauffman et al. (1994) investigate how the ability of local optimisation on an NK landscape is improved if the space is divided into smaller patches. The local changes are accepted or rejected according to the change in fitness of the patch that they are in, regardless of whether other interdependencies in other patches are broken. The authors suggest that this allows the system to escape local optima that would trap the dynamics if system-level fitness was used to determine state change acceptance. They find some improvement in system-level fitness with small patches when systems are rugged (high K, the number of epistatic links for each variable). For smoother landscapes the lowest energy is found when the entire system-level fitness is used to determine state changes.

However the problems that they investigate are on a lattice (*i.e.*, the interdependencies are localised according to nearby sites), and that lattice information is used in splitting the problem into smaller sub-problems. There is no concept of changing multiple variables at once; no possibility of search in combinations of successful module ('patch') solutions.

Overall, this provides support that decomposition can be beneficial in very rugged spaces, provided a decomposition is known *a priori* (or discovered, but this work does not attempt to do so), and even if performed in a reductionist manner: simply optimising each component in isolation, without then considering a higher-order search, or some form of conflict resolution.

2.4.2.2 Cooperative Coevolution

Potter and De Jong (1994) introduced cooperative coevolution (CCEA), a framework that has received much attention. This class of algorithm splits a problem into several sub-problems, and sets a distinct population on each sub-problem: it provides functional decomposition. There is only competition *within* a sub-population, and no competition *between* sub-populations – instead, representative members from each sub-population cooperate to provide a complete solution. Evaluation is collective: a candidate from each population is put forward such that a fully specified solution can be evaluated. Potter and De Jong (1994) use a simplistic pre-defined decomposition with one sub-population for each variable in the problem, an approach followed by several later variants of the underlying CCEA (*e.g.*, Sofge et al., 2002). Selecting exactly which member to be the representative is straightforward when there are no interdependencies between the sub-problems that each sub-population solves, but in the more realistic situation with interdependencies between sub-problems, convergence in each sub-population is an issue. A rather ad-hoc choice of the better out of "best from each population" and "random from each population" is shown to be better than always taking the best. This indicates that diversity is not always adequately maintained in a simple scheme.

Potter and De Jong (1995) consider evolving cascade networks to solve multi-bit parity problems. These networks can be incrementally specified, and when additional nodes are introduced they are assigned to their own sub-population. This method of allocating additional sub-populations may provide traction in specialised domains where solutions of variable length can be evaluated. However, it cannot directly be applied to decomposing a problem that requires a fully specified solution of fixed length. Potter and De Jong (2000) also use a variable number of sub-populations, applying the CCEA to a string matching problem. In this problem, the fittest sub-solution is selected for each of the target strings. Their approach is appropriate for the particular task that requires a general ability that can be decomposed into multiple solutions (see the section below on categorical modularisation), but does not hold in general for an approach that aims to solve parts of a problem that will later be combined.

Shi et al. (2005) recognise that decomposing a system into its underlying dimensions may be disruptive where there are interdependencies between some dimensions. Their solution is to apply a top-down decomposition that splits the problem into two sub-problems of equal size.[2] van den Bergh and Engelbrecht (2004) pre-define a number of sub-populations to split the problem into, "hoping that some correlated variables will end up in the same swarm" (p229). Their allocation uses the variable ordering *a priori* (although the authors do not acknowledge this as an issue or prior assumption).

Finally, Yang et al. (2008) split the problem space into a small number of sub-populations, and randomly guess an allocation of variables to sub-populations. Their approach is more elaborate, as the random decomposition is used to define what each sub-population EA will work on, but also for an auxiliary search that aims to find weights for each sub-population that may repair the overall solution. Furthermore, after each round (where all sub-populations have been evolved), the decomposition is once again re-randomised.

None of these algorithms provide a systematic method to identify suitable decompositions, and are therefore not appropriate unless significant information regarding the system structure is available. It is unclear what advantage can be gained in guessing the decomposition periodically, and while the authors of (Yang et al., 2008) do report good results, it is not clear that the success can be attributed to this feature.

Parker and Blumenthal (2003) use Potter's CCEA framework and investigate different evaluation methods. The trade-off that they consider is that: a) we desire accurate

[2]and presumably allocate dimensions $x_0 \dots x_{n/2-1}$ to the first sub-problem, and $x_{n/2} \dots x_{n-1}$ to the second sub-problem, although it is not stated in their paper.

evaluation, but b) evaluating all members of each population with all possible combinations of the other populations is too expensive. Therefore, how can we achieve the right balance of accuracy and cost? Their empirical study is limited to a robotic box-pushing task, and they find that using somewhere between a single sample and all possible samples provides the best trade-off.

Wiegand et al. (2001) looks more formally at collaborator selection in CCEAs, investigating the size of collaborator pool, how strong selection is in identifying the collaborator pool, and credit assignment. They find that a larger collaborator pool improves matters; and that assigning the fitness from the best collaboration that an individual participated in gives the highest quality optimisation.

Both of these studies look at an issue that we confront in the algorithms introduced later in this thesis: how can we ensure an adequate sample of contexts to test for association formation? In the RSSA (Section 4.2), we generate random contexts and need to use a large sample in order to form appropriate associations. We consider some potential methods for focussing the computational effort. However, we later conclude that using micro-scale search to find locally optimal contexts in order to guide the association formation is a superior paradigm – see Chapter 5.

These two studies also indicate that diversity maintenance is probably necessary when there are multiple solutions for each module (as there will likely be in a multimodal and nearly-decomposable space). If the EA used in each sub-population only has weak selection pressure applied, this could help – but we will use a far more explicit method to maintain different solutions in our approach. The underlying CCEA is nevertheless still dependent on an *a priori* specification of the system decomposition.

Yong and Miikkulainen (2001) evolve controllers for robot tasks, comparing the evolution of one central controller for all robots with a decomposed approach where each robot is evolved in a separate population (similar to Potter's CCEA). In such a domain, it may make sense to decompose the system in this manner, as the inputs and outputs for each robot make obvious decomposition boundaries. However this approach does restrict the applicability to domains where a decomposition is known.

2.4.2.3 Evolving Architecture and Sub-Solutions Separately

In a somewhat different category of coevolutionary decomposition methods, some algorithms separate the task of solving sub-problems from arranging the particular combination of sub-solutions in any one system. These are more often applied to network-type tasks than combinatorial optimisation, where there is some fungibility in the specific tasks undertaken by each sub-component.[3] Taking such an approach may restrict the (flex)ability of a system to automatically adjust its architecture, pre-supposing a

[3]Khare (*e.g.*, 2006) labels these 'two-level coevolutionary methods'.

two-level architecture; however the ability for sub-solutions to themselves specify sub-architectures would regain such flexibility. Further, the distinction is not clear-cut. If bottom-level components can vary in size and/or task that they fulfil, then the system architecture can evolve in a decentralised manner.

An early two-level method is Husbands and Mill (1991) work that takes a coevolutionary approach to a decomposed (planning and scheduling) problem. It is similar to Potter's CCEA, having several sub-populations that each address different sub-problems. However, one additional sub-population consists of arbitrators, which resolve conflicts over resources between the other sub-populations, doing so in a manner that suits the overall system quality. This work also explicitly specifies the system decomposition beforehand: the number of process plans, and their individual targets are all known *a priori*.

Moriarty and Miikkulainen (1998) evolve neural networks for classification tasks using a two-level approach. The lower-level units are neurons, which are composed into systems by a second evolutionary process on networks. There is no manner in which this work can recursively compose higher-order sub-solutions from lower-order sub-solutions: only one layer of decomposition is possible. Khare and Yao (2004) take a very similar approach, modifying the exact type of neuron employed from perceptrons to radial basis functions. In both of these pieces of work, there is no specific task allocation to any neuron group: all neurons can be employed at any position in the network.

Khare et al. (2005) present a method for automatic problem decomposition for time series prediction. Their architecture however is contrived: it uses two distinct populations with pre-defined roles. First are the 'module' candidates that actually do the problem solving, and second are the 'system' candidates, which combine the outputs from exactly two modules. This successfully identifies suitable inputs to each module to decompose a two-component task.

2.4.2.4 A Note on Categorical Modularisation

We discussed different types of modularity in Section 2.1.3, identifying functional decomposition as our primary interest, in contrast to any assumption that regularities exist in the search space. Both regularity, and functional decomposition, are properties of the problem or system. Some techniques attempt to find decomposition in the solution that they provide. For instance, Darwen and Yao (1997) take a 'categorical modularisation' approach to evolve sets of strategies for the iterated prisoner's dilemma. Each strategy could in principle be played against any opponent, but the approach taken here is to provide a set of specialised strategies rather than a single generalised solution (and accordingly, this approach must also design a mechanism to determine when to deploy each strategy).

This approach can be seen as a form of 'bet-hedging' (Beaumont et al., 2009; Worgan and

Mills, 2008), where the tasks exhibit some form of modularity, but temporally rather than functional groupings within the same static task. In our approach, maintaining multiple different sub-solutions to each sub-problem is very important on account of context dependency. It is not the case in general that the solution to one sub-problem could stand in as a solution any other sub-problems. Thus, categorical modularisation is a fundamentally distinct approach from functional decomposition.

2.4.3 Restrictions of the Current Approaches

Throughout this section, we have identified a number of limitations with current decomposition methods. We discuss two in particular: the use of prior information; and the maintenance of diversity.

Use of Prior Information. Most of the methods discussed above use some knowledge of the system structure, architecture, or decomposition. Those methods that do not presuppose such knowledge instead guess a decomposition (*e.g.*, Shi et al., 2005; Yang et al., 2008). While either approach may provide traction for the specific domains tested, these do not provide a principled manner in which to identify the structure of a system in order to solve sub-problems efficiently. Moreover, using structural information restricts how well techniques can transfer to a space where very little prior information is available.

Finally, pre-specifying a decomposition or an architecture reduces the flexibility of the system. A fixed architecture precludes the possibility of refining the initial decomposition as an algorithm runs, and in particular, recursion (where evolving higher-scale structures from lower-level components is available more than once).

Diversity Maintenance Potter's work is interesting and has stimulated much further research. However, at least one difficulty with his method is that we may not know which of the possible options in a sub-population is most appropriate. Classical EAs are known to suffer from premature convergence, hence the significant effort to develop niching and other diversity maintenance techniques. Yet maintaining multiple possible sub-solutions is critical when there is modular interdependency – *i.e.*, the best solution depends on the context. Much, if not all, of the work reviewed above overlooks the same issue – appealing to Potter's explanation that isolated sub-populations automatically provides diversity maintenance – but this is not the case *within* a sub-problem. Moriarty and Miikkulainen (1998) do acknowledge the issue of diversity (as central to problem decomposition), and employ implicit fitness sharing in their method to promote diversity. Since the sub-components can all potentially stand in for one another, the question is slightly different: Potter's approach does not assume that one component can act as a solution for anything but the sub-problem for which the sub-population that it came from was assigned to solve.

2.5 Compositional Optimisation

The Symbiogenic Evolutionary Adaptation Model

SEAM is an evolutionary algorithm inspired by symbiogenesis (Watson and Pollack, 2000, 2003). It differs from typical evolutionary algorithms in several aspects. These include the symbiogenic variation operator, an ecosystem of entities that are partially specified, and coevolved ecosystem templates that are used to facilitate evaluation. The main pool of genetic material represents an ecosystem of many different entities, each specifying only part of an overall solution. In order to allow fitness evaluations of these partially specified entities under black-box optimisation assumptions, contexts are generated from other members of the ecosystem. There is no source of new genetic material (*i.e.*, no mutation), and the only variation operator is the symbiotic join of two entities.

The main loop of the algorithm operates as follows: two entities are picked at random, and evaluated in a number of contexts to determine if the pair should make a permanent symbiogenic alliance. A version of Pareto-dominance is used to make these decisions. Since joins are irreversible, the rationale behind join decisions is to be conservative. Therefore, the proposed composite must offer an improvement to the constituent entities over free-living in at least one context, and a detriment in none. That is, the join must Pareto dominate free-living, or else no change to the population occurs. When joins are made, the two symbionts are removed from the ecosystem and replaced with the chimera. As this process is repeated, the average size of entities will increase until fully specified solutions are discovered.

SEAM introduces a powerful new concept to evolutionary computation: compositional search. However, it has several limitations: the Pareto dominance mechanism employed to determine whether a join should be made is unnecessarily complicated. Assembling contexts from other entities within the ecosystem, although arguably natural, is an unnecessary assumption, which we will demonstrate in Chapter 4. Finally, SEAMs performance in other problem domains is brittle: the utility of SEAM has only been demonstrated on a stylised hierarchical function, Hierarchical If-and-only-If (HIFF, Watson et al., 1998).

The Hierarchical Genetic Algorithm

The hierarchical genetic algorithm (HGA; de Jong et al., 2005b) explicitly utilises the concept of modular interdependency in a manner inspired by SEAM. It attempts to reduce the search space for a given subset of variables by determining which of the possible settings for that subset are optimal in some context. If a particular setting never demonstrates that it is optimal for any context, it is no longer considered. Larger modules are formed when variable settings are discarded, and this process is repeated to result in modules containing only global optima.

The HGA is closely related to SEAM, and shares the limitation that it has only been demonstrated to operate successfully on hierarchically decomposable problems.

The Evolutionary Transition Algorithm

The Evolutionary Transition Algorithm (ETA Defaweux et al., 2005; Defaweux, 2006) incorporates a symbiotic variation mechanism into a more traditional evolutionary algorithm framework that maintains a population of partial solutions. When selected for representation in the next generation, the partial solutions can potentially undergo composition to become a larger partial solution. ETA was developed specifically for binary constraint satisfaction problems, a class of problems that permit partially specified solutions to be evaluated. Allowing partial evaluation changes problem characteristics, when contrasted with black-box optimisation assumptions. Specifically, some problems that are difficult under black-box optimisation assumptions become easy when under partial evaluations. For instance, finding ground states of a lattice-based Ising model presents many local optima when all variables must be specified. However, if it is possible to specify variables incrementally, we can resolve the dependencies according to only the variables that are currently defined to straightforwardly find a global optimum.

Taking some ideas from both the ETA and MACRO approaches, we could modify the algorithms in one of two directions. First, we could apply our methods to domains where using partial evaluation is appropriate. Second, we could incorporate a separation of timescales between composing new units and using those units into the approach of ETA. Either of these could lead us to a broadened understanding of the influence of such constraints on compositional techniques.

2.6 Conclusions

In this chapter we have described several different evolutionary approaches to computational problem solving, focussing in particular on methods that aim to gain traction by decomposing a problem into smaller sub-problems. The approach proposed in this thesis is distinct from all of material reviewed above, but elements of the algorithms described above do strongly influence our design. In particular,

- Collective problem solving: different parts of a problem are solved by different members of the population, rather than attempting to maintain a population of full solutions;
- Encapsulation of lower-level components: once a lower-level solution is found, future optimisation is performed at a higher scale, using that sub-solution as an abstract component;
- Recursive application of encapsulation;
- Discovery of epistatic interactions through model building.

In Chapter 5 we introduce the first implementation of MACRO, which explicitly encapsulates lower-level units into composite units in a recursive manner. The compositional algorithms (Section 2.5) and the recursive DSMGA+ (Section 2.3.3) are the most similar methods to this approach. In Chapter 6 we introduce a more elaborate implementation of MACRO, which uses probabilistic associations between variables (see also the third implementation in Chapter 7). The model building used by our probabilistic algorithm is most closely related to BMDA (Section 2.3.2).

The two most fundamental differences that our approach has from the algorithms detailed in this chapter are the manner in which we employ the probabilistic models, and the separation in timescales between creating macro-scale variational units and performing search with those units. In particular, EDAs draw new fully specified candidates from their models. This is in contrast to our approach, which uses the probabilistic model to define macro-scale variational units that are used for higher-order hill climbing. These units only specify a partial solution in general. Crucial to our approach is adapting the model of system decomposition on a slower timescale to the hill climbing process that operates using this decomposition. This underpinning concept distinguishes our approach from both EDAs and existing compositional search methods.

Chapter 3

The Variable Structural Modularity Problem Representation

Systems from many different domains exhibit modularity, from phenotypic development in biological systems (Wagner and Altenberg, 1996) to human-engineered systems where, for example, the concept of decomposability is one of the key assumptions required for cell library utilisation in ASIC design (Smith, 1997). In computational problem solving the concept that one portion of a system can be solved in partial or full isolation from other portions of the system is both simple and intuitive, and forms the basis for the building block hypothesis (BBH) (Holland, 1975, 2000; Goldberg, 1989; Mitchell et al., 1994). However, identifying problems that exhibit modular properties of the right kind to allow solutions to be more easily accessible to algorithms that exploit these modular properties is not straightforward (Forrest and Mitchell, 1993b).

We aim to better understand the algorithmic properties that can and cannot overcome difficulty introduced by modularity. The methodology that we take in pursuing this broad aim is to use problems with known structure and solutions, in order that we can determine the factors that are critical to the performance of these algorithms in a principled manner. In particular, we are interested in a test problem that satisfies the following three requirements: 1) a principled construction to enable straightforward interpretation; 2) flexibility in the strength of modularity created, such that we can investigate the changes that occur in problem solving ability when the structures that created modularity are blurred; and 3) it should allow epistasis between modules (*i.e.*, the modules should not necessarily be separable). Thus, we arrive at the specific research question that we aim to address in this chapter:

How can we represent modularity using a flexible and natural approach?

This chapter describes a new approach to representing modular problems, which parameterises the amount of modular structure that is present in the epistatic dependencies of the problem. The representation includes epistatic interdependencies between each pair of variables, which each make a contribution to the overall system fitness. The interaction can be resolved if the variable configurations are appropriately aligned or anti-aligned (depending on the sign of the interaction). The key to making our problem variably modular is very simple. We define the fitness of a genotype using a sum of weighted pairwise interactions between the problem variables. For the highly modular case, strong interaction weights within modules are cleanly divided from the weak interactions that are between modules. In the non-modular case these weights are randomised so that weights inside modules are not different on average from weights between modules. A partially modular problem is defined by partially randomising the weights of the modular case.

When the important dependencies are structured into tight clusters, the problem landscape can exhibit significant ruggedness. Each module has multiple optima, and the fittest configuration depends on the context of the rest of the system: the system exhibits modular interdependency (Dauscher et al., 2006). Therefore, a mechanism that cannot search in the space of module configurations will not be able to efficiently resolve all of the interdependencies – but a mechanism that can effect appropriate modular variation will in principle be able to reliably solve the problem. Conversely, when the clustering of strong dependencies is towards the non-modular end of the scale, the ruggedness is reduced and simply following local fitness gradients is sufficient to solve the problem.

This problem representation advances our conceptual understanding of modularity in the following two main aspects:

1. Constraints that act in concert create modules, and even a weak pressure to align module solutions results in local optima; but the structure introduced can be resolved by mechanisms that search with appropriate modular variation.

2. As the clustering is 'tuned out', local optima may be reduced in significance and hence become resolvable for simpler algorithms. This suggests that if we see the type of structure introduced here elsewhere, it may be exploitable by appropriate modular variation.

The ideas in this chapter are published in part in (Mills and Watson, 2007b).

3.1 Prior Tuneable Test Problems, and Definitions

There are many different ways that a problem could be tuned to show a variable amount of difficulty for mutation (*e.g.*, Kauffman, 1993), and many parameters may be involved in the definition of functions that might influence the discrimination of mutation and crossover algorithms (Pelikan et al., 2006a; de Jong et al., 2005a). If nothing else, one could vary the size of a problem or the width of fitness valleys incorporated in the problem (*e.g.*, Jansen and Wegener, 2005; Watson, 2004). One notable factor influencing the effectiveness of crossover is the tightness of the genetic linkage (the defining length of building blocks; Holland, 1975; Harik, 1997). But none of these methods directly addresses a variable amount of modularity in a simple tight-linkage building-block problem.

A modular problem without inter-module interdependencies is defined as separable (Watson and Pollack, 2005). Simon calls particular types of non-separable modular systems 'nearly decomposable' (Simon, 1969). Watson refines this definition to describe problems that are decomposable but not separable as exhibiting 'modular interdependency' (Watson and Pollack, 2005; Dauscher et al., 2006). We do not wish to restrict our scope to separable problems, so we allow inter-module dependencies in our problem. We introduce a method that is intended to describe problems that exhibit modular interdependency. Note that in principle, systems that exhibit modular interdependency may still be easy for crossover (Watson and Pollack, 2005). Note also that in a function built only of pair-wise dependencies, inter-module dependencies must be weak when compared with intra-module dependencies since the strong internal dependencies are what define the module. For the sake of clarity, we shall refer to intra-module and inter-module dependencies as internal and external dependencies respectively. In the following sections, as throughout this thesis, when describing mutation we refer specifically to incremental processes such as point mutation, and not sequence based mutations (*e.g.*, inversion, translocation) unless explicitly indicated.

It should be noted that problems built of pairwise interactions, or only order-2 dependencies, can often be easy to solve (Kauffman, 1993; Goldberg, 1989). But this depends on how the interactions are organised. When dependencies are structured, pairwise dependencies can 'act in concert' to create local optima with significant Hamming distances between them (Dauscher et al., 2006; Watson, 2006). Varying this structure is sufficient to vary the problem difficulty.

We approach the definition of our parameterised modular problem by initially considering a very simple non-modular problem and subsequently incorporating modular structure. We defer the introduction of components required for parameterisation until the basic modular function is established.

3.2 Constructing Modularity

To address the aims outlined above, we define interaction structures using a spin glass representation that provide the desired properties. While it is traditional to consider

Table 3.1: Equivalence Between Ising Spin and Binary String Representations

x_0	x_1	iff (x_0, x_1)	σ_0	σ_1	$\sigma_0 \cdot \sigma_1$
0	0	1	-1	-1	$+1$
0	1	0	-1	$+1$	-1
1	0	0	$+1$	-1	-1
1	1	1	$+1$	$+1$	$+1$

energy minimisation in spin glass systems, in evolutionary systems it is often more natural to maximise fitness. Throughout this thesis, we use terminology that considers function maximisation. We also follow the evolutionary optimisation convention with variable allocations. Ising spin systems use variables $\sigma \in \{-1, +1\}$, whereas in evolutionary computation it is more common to use allocations of $x \in \{0, 1\}$. We use strings of binary variables $\boldsymbol{x} = x_0, x_1, \ldots, x_{n-1}$ by default, but where appropriate we also use spin systems denoted by $\boldsymbol{\sigma} = \sigma_0, \sigma_1, \ldots, \sigma_{n-1}$. The mapping between the two systems is straightforward:

$$\sigma = 2x - 1, \tag{3.1}$$

$$x = \frac{\sigma + 1}{2}. \tag{3.2}$$

The test problem class introduced in (Mills and Watson, 2007b) used the dyadic logical function if-and-only-if (IFF), whose output value is high when both input variables are equal. In an Ising spin system, this function is implemented as the product of two spins – see Table 3.1.

3.2.1 A Simple Unstructured System

We take a system of N binary variables and order N^2 pairwise interactions, whose variables can be assigned values $x \in \{0, 1\}$. Each of the interactions defines a fitness contribution according to the strength of the interaction, that is either positive or negative depending on whether that dependency is resolved.

We can define a simple, unstructured system in which all variables have dependencies with all other variables of equal value $w > 0$. The overall fitness is defined as the sum over all pairwise dependencies as defined in Equation 3.3.

$$F(\sigma_0, \sigma_1, \ldots, \sigma_{N-1}) = w \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sigma_i \sigma_j. \tag{3.3}$$

Since w is positive, when variables have agreeing values, the dependency between those variables is satisfied and the fitness contribution is positive. Conversely, the contribution is negative when variable values disagree. This is the function of equality.

Although this system contains many dependencies, it is not structured. There are two optima in its fitness landscape, each of equal fitness. Paths of monotonically increasing fitness exist that lead from any initial configuration to one of these solutions, and therefore even the simplest adaptive process would find a global optimum.

3.2.2 A System with Modular Structure

To introduce structure into this system, we allow each interaction to make a contribution of different strength, instead of assuming that all interactions are of equal importance. We use two classes of weights: w_I as the strength of interaction between variables that are within a module, and $w_E \leq w_I$ as the strength of interaction between modules.

Using just two weight classes keeps our problem formation simple, and yet allows us to represent a number of modules within which the internal dependencies are far more important than those outside of the module. Structuring the strong internal interactions close to the leading diagonal introduces a number of modules. We define Z modules of k variables such that $Zk = N$. Equation 3.4 specifies the positioning of the weights (see also Figure 3.2 (a)):

$$w_{ij} = \begin{cases} w_I, & \text{if } \left\lfloor \frac{i}{k} \right\rfloor = \left\lfloor \frac{j}{k} \right\rfloor, \\ w_E, & \text{otherwise.} \end{cases} \tag{3.4}$$

The overall fitness is therefore given by eqn 3.5.

$$F\left(\sigma_0, \sigma_1, \ldots, \sigma_{N-1}\right) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sigma_i w_{ij} \sigma_j . \tag{3.5}$$

Because both w_I and w_E are positive, dependencies are satisfied when variable values agree. However, local optima are created when all within-module dependencies are satisfied, but not all between-module dependencies are satisfied. This results in a fitness landscape that has significant ruggedness. We examine a 20-module example in Section 3.2.3. Note that it is not required that all modules are of equal size, but this simplification is suitable for our purposes.

It is possible to create a functionally equivalent system with weighted fitness contributions defined by the Boolean subfunction if and only if (iff). Watson (2006) takes this approach to define a version of the Hierarchical-IFF function built from pairwise dependencies. Here we address a more straightforward two-level modularity rather than hierarchical modularity. This concept of modularity represented by a pairwise dependency matrix with high values grouped along the diagonal in this manner is both natural and intuitive (Simon, 1969; Lipson et al., 2002; Segré et al., 2000; Higgs, 1996, 2000; Morgan and Higgs, 1998; Watson, 2006; Sporns, 2006; Page et al., 2007).

The strong internal dependencies provide a selective gradient towards all variables within

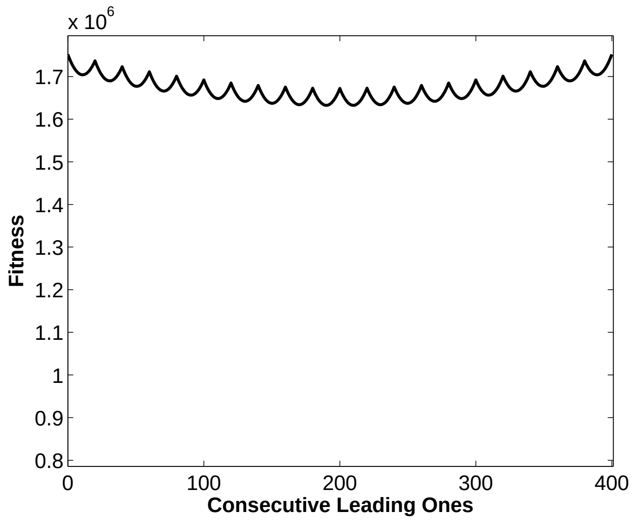


Figure 3.1: A cross section through the fitness landscape for the clean VSM problem with N=400, Z=k=20, w_I=200, w_E=1.

a module agreeing, resulting in two equally fit optima that are equally easy to find. As such we expect the probability of finding each optimum of a module to be on average 0.5 when considered in isolation. However, the overall fitness of a genotype will be increased in proportion to how many pairs of modules 'agree' in how they are solved, *i.e.*, if either both are at the all-down solution or both are at the all-up solution. This is because these configurations also confer the fitness contribution bonuses from the external dependencies. Therefore the optimal solution to a module is not independent of context, but is nevertheless always one of only two possibilities, all-down or all-up. This modular interdependency (Dauscher et al., 2006) requires a search method that explores combinations of modules in order to find optimal solutions – in later chapters we shall explore some mechanisms that can provide such an exploration. Specifically, in Chapter 4 we investigate when crossover operators can explore combinations of modules, and in Chapters 5, 6 and 7 we investigate symbiosis-inspired mechanisms that learn module structures and consequently search at the module level.

3.2.3 An Example System, and Properties

We consider a specific landscape for the function described by Equation 3.5 and Equation 3.4, for $N = 400$, $Z = k = 20$, $w_I = 200$ and $w_E = 1$. Figure 3.1 shows a particular cross-section of the fitness landscape defined by this function. The curve shows the fitness of each genotype $\boldsymbol{x}$ from $\boldsymbol{x} = (0)^N$ to $\boldsymbol{x} = 1^N$ and all genotypes $\boldsymbol{x} = 1^i\,(0)^{(N-i)}$ in between, *i.e.*, 00000. . . , 10000. . . , 11000. . . , 11100. . . etc.

This slice through the fitness landscape shows 21 different optima at 11 different fitness values. In the entire landscape there are 2^Z local optima: the sparseness of optima is $2^Z/2^N = 1/2^k$ ($\approx$ 1 in 10^6). However, each of the 21 optima shown in Figure 3.1 have

$\binom{Z}{X}$ equivalents, where X is the number blocks solved with all-1s at that point in the section. Although this cross-section only shows $N+1$ of a possible 2^N genotypes, all of the local optima in the landscape are either on or have an equivalent on the cross-section, and so it provides valuable insight into the formation of the landscape. In Appendix A, we derive the positions of the optima, and hence show that there are 2^Z optima in the space.

One of the properties that we want the VSM problem to exhibit is ruggedness sufficient to prevent a mutation-only protocol from finding a global optimum. That is, the expected time for a protocol that uses only uncorrelated changes to solve the problem will scale exponentially with the problem size.[1]

In addressing this, we should consider what makes the problem difficult. First, the ratio between w_E and w_I controls the presence of local optima, and their height relative to nearby configurations. Second, assuming a weight ratio that ensures all potential local optima are present, increasing the number of modules leads to an exponential increase in the number of local optima for a given problem size. Third, increasing the module size leads to widening fitness valleys that separate local optima. In the example above we set $Z = k$ to maximise the trade-off, but it is only important that both the number of modules, and the module size scale with problem size.

Let us now explore the effect of weight ratio on these three aspects. As the internal and external weights tend to equality, the local optima disappear. In the limit of $w_I = w_E$ the function becomes the simple problem of maximising the majority of 0s or 1s (as mentioned in Section 3.2.1). At the other extreme, when w_E/w_I tends to 0, the difference in fitness from lowest to highest optima also tends to zero. In this case, the width of the fitness valley between each local optimum and neighbouring local optima is at its maximum.

However, there are intermediate weight ratios where there are 2^Z optima, and the global optima are significantly fitter than other local optima.

In Appendix A, we derive thresholds for the weight ratio at which each class of potential local optima disappear (when increasing w_E relative to w_I). The resulting equation is given in terms of k, Z, and X, where X is the majority number of modules that are solved to the same value (*i.e.*, the greater of all-0 or all-1 modules):

$$\frac{w_I}{w_E} = \frac{2Xk + k - Zk}{k-1}. \tag{3.6}$$

The threshold for the class of optima that have $Z-1$ modules solved to the same type is the highest, and in the example above this is at 20:1. For the examples in this chapter,

[1]It is also important that there are plausible mechanisms that scale polynomially with the problem size. Such mechanisms are investigated in later chapters.

we choose $w_I : w_E$ to be above this threshold at 200:1 which provides a width of 18 bits for the last fitness saddle. This saddle width is not the potential maximal 20 bits that it would be if the modules were separable. The discrepancy indicates the influence that the external weights have.

At a ratio of $w_I : w_E \to 0$, an algorithm that makes changes of fewer than k bits will find the optimum closest to the initial condition it starts from. Once at a local optimum, the k bit change required to move to a higher fitness optimum is unavailable. Therefore, such an algorithm will only succeed if it samples an exponential number of basins, and scales as $\mathcal{O}(2^Z)$.

If a k-bit mutation rate is used, the possibility of moving between local optima exists. However, in order to change exactly the correct k bits and not modify any of the otherwise correct bits, the single specific event must occur from the full set of $2^k\binom{N}{k}$ different events. This scales exponentially with the problem size when $k \propto N$.

Therefore, the ruggedness in the VSM landscape presents significant difficulty for any algorithm that uses only uncorrelated changes.

3.3 Parameterising Structural Modularity

It still seems restrictive to only consider such neatly formed modular problems, even if they do exhibit modular interdependency. We can modify the problem in a number of ways that will vary the level of modularity that is present whilst still maintaining the essence of the ideal modular problem, at least for a portion of the variation scale. These include:

- Variation in the values of weights;
- Variation of the position of the classes of weights; and
- Variation of the function that satisfies dependencies.

Varying the problem using these different methods can modify the properties in significantly dissimilar ways; this is discussed further in Section 8.3. In this section we investigate a single method of adjusting the modularity present: to modify the positions of the classes of weights in the system. Note that varying the correspondence of the epistatic modularity to the genetic map (*i.e.*, varying the tightness of the genetic linkage) is another way to vary the ease with which a genetic algorithm will find good solutions. However the correspondence of epistatic modularity with the genetic map is a different issue from varying the modularity of the problem *per se*, as addressed here: our

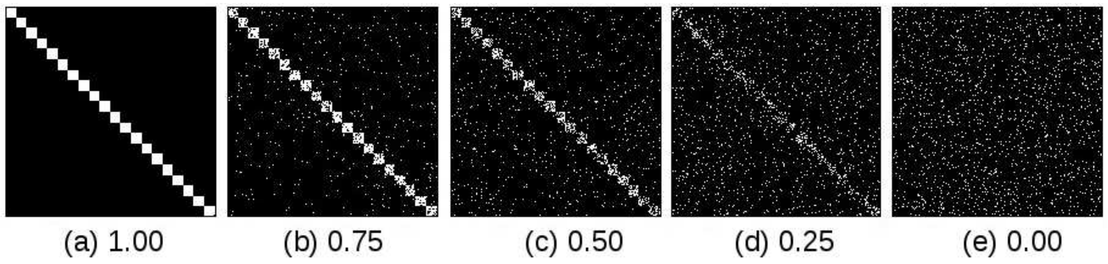


Figure 3.2: Example weight matrices for N=400, Z=k=20, w_I=200, w_E=1 for a range of modularity levels. Note that the dark region indicates the value w_E, which in general is nonzero

method of varying the modularity in the problem is not equivalent to shuffling the genetic map as it is not guaranteed that any amount of rearranging the loci would recover perfect modularity.

In order to explain the scale we use for structural modularity, we first consider the limits before deriving equations to describe the distribution of weights for the parameterised case.

Figure 3.2 (a) shows how the weights are organised for a neatly modular VSM, with the locations of internal weights w_I in white, and external weights w_E in black. As we decrease the level of modularity, we start to see external weights where internal weights once were, and vice versa. We label the region where internal weights exist in the neat VSM as r_I (*i.e.*, the block diagonal), and similarly use r_E for the region containing external weights. Figure 3.2 (b)–(e) shows the organisation of weights for a range of levels of structural modularity.

Regardless of the level of modularity, the number of w_I weights and w_E weights remains constant so as not to change the total amount of epistasis in the problem. Thus, the proportions of the total number of weights that have the values w_I and w_E are $1/Z$ and $(Z-1)/Z$ respectively.

In the neat VSM problem (i.e. at structural modularity=1), we see the proportion of w_I weights in region r_I is 1, and correspondingly find the proportion of w_I weights in r_E to be 0. We define the other end of the scale to have w_I weights randomly distributed, and as such expect to find a density of $1/Z$ in both regions r_I and r_E. For intermediate points in this scale of structural modularity, we linearly interpolate the proportion of w_I weights in r_I from 1 to $1/Z$, and the proportion of w_I weights in r_E from 0 to $1/Z$. Specifically, for a level of modularity ρ, the proportion of weights in r_I that have the value w_I is $1/Z + \rho(1 - 1/Z)$, and the proportion of weights in r_E that have the value w_I is $1/Z - \rho(1/Z)$.

Figure 3.3 shows the same cross-section through the genotype space as Figure 3.1 all of the genotypes $\boldsymbol{x} = 1^i\,(0)^{(N-i)}$, for instances with varying levels of modularity, from ran-

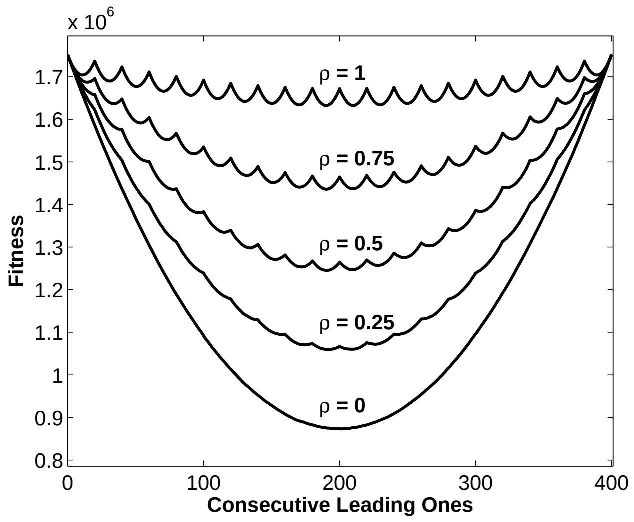


Figure 3.3: A cross section through the fitness landscape for the VSM problem with N=400, $Z = k = 20$, $w_I = 200$, $w_E = 1$, with varying degrees of modularity, ρ. As the modularity increases, these cross-sections suggest an increase in the height and number of local optima in the problem

domly distributed w_I weights (ρ=0) to neat structural modularity (ρ=1) in increments of 0.25 (see also Figure 3.2 (b)–(e)). In the non-randomised case (ρ=1), we formally characterised where the positions of local optima are (see Appendix A), and can be confident that the particular cross-section selected is meaningful and informative. Since the positions of weights are partially randomised in these landscapes, we must express caution when interpreting the information provided by the one-dimensional cross-sections. The landscape cross-section for ρ=0 indicates an absence of local optima. As we increase the structural modularity towards ρ=1, the landscape cross-sections suggest an increase in the number and height of local optima. Note that as all dependencies have positive sign, the global optima are always at all-down and all-up, regardless of the level of structural modularity.

3.4 The Relationship Between Local Optima and Weight Ratio

In the previous section, we investigated the relationship between the ruggedness of the VSM landscape and the strength of structural modularity. Here, we examine the landscape properties as the ratio between internal and external weights is varied, while keeping the structural modularity as strong as possible (*i.e.*, ρ=1).

We use a simple hill climbing algorithm to provide information as to the ruggedness of the landscape. Specifically, we use a restart single-bit-flip hill climber. This algorithm modifies exactly one locus in each generation, and accepts the change if the fitness of the

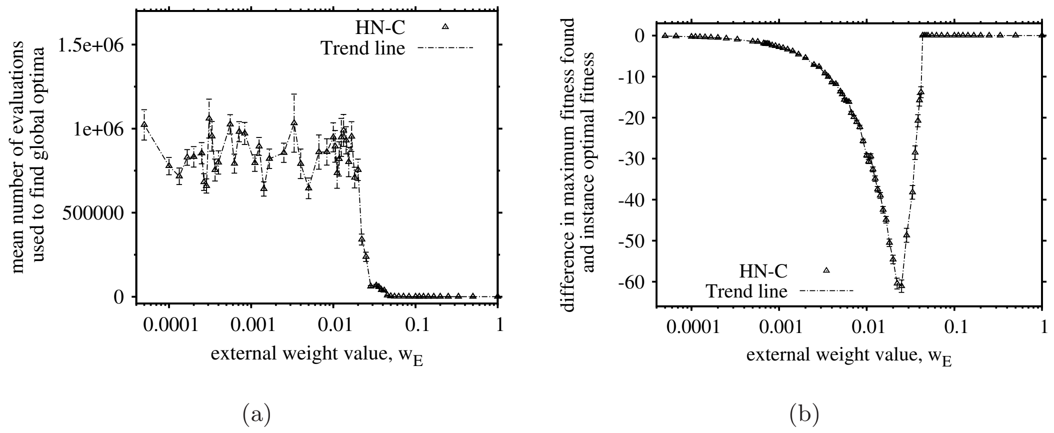


Figure 3.4: How the VSM ruggedness depends on weight ratio. (a) number of evaluations required for a hill climber to find global optimum; (b) maximum fitness found after 54 restarts. In both graphs, data points are connected to indicate the overall trends.

new state is non-worse. The hill climber runs from a random initial condition for a pre-defined number of evaluations, τ, and then restarts at a new random initial condition. This algorithm provides some empirical measurements of how rugged the landscape is at the micro-scale. As the algorithm only modifies one variable at a time, the local optima identified in the discussions above are truly inescapable.

For these experiments, we set τ to $\lceil \frac{3}{4} \cdot T \rceil$, where $T = eN \ln N$. T is from the result of Mühlenbein (1992) that $T = e^{\mu} N \ln N$ is the expected time for a mutation algorithm to solve *onemax*, where μ is the expected number of mutations per timestep, and N is the number of variables in the problem ($\mu = 1$ for this algorithm). By inspection, we find that the system has reached a local optimum within $0.75 \cdot T$ in almost all cases.

We perform two experiments using this restart hill climber. In the first experiment we measure the number of fitness evaluations taken to find a global optimum. In the second experiment we measure the highest fitness discovered after a fixed period of search. The parameters used here are: k=5, N=100, Z=$\frac{N}{k}$=20, $\tau = 939$. By allowing a computational budget of approximately 50,000 evaluations, we arrive at $\lceil \frac{50{,}000}{939} \rceil = 54$ restarts. For these experiments, we set w_I=1 and vary w_E. We test a range of ratios between 1 and 20,000 (*i.e.*, $1/20,000 \leq w_E \leq 1$). These parameter values illustrate the properties of VSM problems, and are by no means required to demonstrate the qualitative results of Figure 3.4.

3.4.1 Ruggedness in the VSM Landscape

The results of the first experiment are shown in Figure 3.4 (a). We observe three main portions to the response of the restart hill climber. First, at high weight ratios, the hill climber takes very few evaluations to find a global optimum, indicating that there are very few or no local optima. Second, at intermediate ratios, the number of evaluations taken transitions fairly rapidly from under 1000 to approximately 10^6. Third, at low ratios, the number of evaluations taken is fairly constant at approximately 10^6.

Let us consider the two extreme portions first. At high ratios, the internal weights are not strong enough to create local optima (not counting the two global optima). Consequently, the hill climber finds a global optimum within one restart. Below a certain point in the weight ratio, all of the potential local optima in the problem are present in the landscape. Therefore, many restarts are required to sample a basin of attraction that leads to a global optimum. However, decreasing the weight ratio further does not introduce any additional optima, and so the number of evaluations required does not systematically increase.

The transition from easy to hard is fairly rapid. The general trend in the problem structure is that as the weight ratio is decreased, more local optima are introduced. This is reflected by the increase in the number of restarts that are required to find an initial condition in the right basin of attraction. Let us now refer to Equation 3.6, reproduced from Appendix A. This equation describes the thresholds in weight ratio at which each class of potential local optima appear (with decreasing ratios). For the problem parameters in this instance, the ratio at which the last local optima class appear is at 1.25:1, leaving only the two global optima. All of the potential local optima classes are present when the ratio is lower than 23.75:1. Our empirical data suggests that the mean time to find a global optimum is less than one restart for ratios of 14:1 and above, while for ratios below 50:1 the number of evaluations has stopped increasing. The analytical equation describes when the local optima appear/disappear, but does not formally describe the sizes of the basins of attraction for each class of optima. At ratios $\rightarrow \infty$, the basins tend to having equal size. Conversely, when there are only two classes of optima, the global optima have a significantly larger basin of attraction than the local optima. Thus, we would expect for the basin of attraction of the global optima to continue to decrease as the weight ratio is decreased below the exact ratio of 23.75:1.

3.4.2 The Quality of Global Optima

In the first experiment, we saw that as the weight ratios decrease, the problem ruggedness increases. Once all of the local optima are present and the basins have approximately the same size, the time taken for the restart hill climber flattens off. However, what the first experiment does not reveal is the relative quality of solutions found compared to the global optimum. Our second experiment provides some information on this aspect. The results in Figure 3.4 (b) show the highest fitness discovered by the restart hill climber within 54 restarts across a wide range of weight ratios. The mean from 100 independent repeats is used for each data point.

These results reveal that there is an inherent trade-off between problem difficulty and relative fitness of global optima to local optima, controlled by the weight ratio. Specifically, at high weight ratios (as $w_I/w_E \to 1$), the within-module dependencies are not strong enough to differentiate the modules. Under these conditions, the restart hill climber finds a global optimum reliably. As the weight ratio decreases, the relative fitness drops rapidly, to a low point at a ratio of 40:1. As the ratio decreases beyond this point, the relative fitness begins to increase again.

At the other end of the spectrum, where $w_I/w_E \to \infty$, the between-module dependencies become almost entirely irrelevant. Specifically, at a ratio of 500:1, the highest fitness discovered is within 1% of the globally optimal fitness, and by 10000:1, within 0.1%. If optimising for a high-fitness solution rather than the global optimum, any local optimum would be almost as fit as the maximum fitness configuration. Here, the problem has become almost separable. Thus, as the ratios move from high to low, the difficulty increases, but the relative value of finding a global optimum decreases.

The experiments above illustrate that in our system, at intermediate weight ratios, solving the system can be both difficult (*i.e.*, the landscape is very rugged) and significant (*i.e.*, the fitness of the global optima is meaningfully higher than the mean optima fitness). This could be characterised as the point at which modular interdependency is strongest – although a direct labelling of the weight ratio axis as modular interdependency is not appropriate.

3.5 Discussion

We highlight two idealisations made by the VSM problem construction. Each of the modules have a regular composition: they are identical in size, and the configurations that have maximal fitness are the same for all modules (all-0 and all-1). Assigning exactly k variables to each module might give an algorithm such as macromutation (Jones, 1995a) a better chance of solving the problem if k were known – it would reduce the search from having to find a) the starting location of a module; b) the size of the module; and c) an optimal configuration to only needing a) and c). However, since we define k to scale with the problem size, this approach would still scale exponentially. Since the modules all have the same solutions, and further, fitness is maximised when the module solutions all agree, there is some regularity that could be exploited. However, any mechanism that copies successful loci allocations from one part of the genome to another (*e.g.*, translocation or transposition (Voss and Foley, 1999; McGregor and Harvey, 2005)), whether entire modules or single loci, would be trivially defeated if the target in each module were put through an XOR operation with a random target. None of the algorithms developed in this thesis use any information regarding regularity in this or other test problems – to do so would be to misunderstand what the problems were designed to test.

The structural organisation that we define in the VSM problem is all regarding epistatic linkage (*i.e.*, the functional relationships between variables in the system). A second class of organisation is physical linkage, which describes the positional relationships between the variables in the system. To obtain this information would require expert labelling in most systems, and is in general unknown. Therefore, most of the investigations in subsequent chapters in this thesis make no assumption regarding any correlation between epistatic and physical organisation. The one exception to this is in the first part of Chapter 4. There we investigate a genetic algorithm that uses linkage-preserving crossover, and, without assuming a tight correspondence, this work would reveal little with regards to the exploitation of modularity.

The non-modular end of the structural modularity scale in the VSM is trivial even for simple algorithms because there are no conflicting constraints. It is not the case that the system is entirely separable, or unconstrained, but because resolving one constraint does not break any other constraints. Introducing modularity to the system makes it more difficult for local search algorithms on account of the increase in frequency of local optima. As we shall see in later chapters, if an algorithm can provide variation that is appropriate to handle the modularity at its strongest, then the strength of modularity does not impact upon its success rate.

Compare this with an initial system with no structuring in its interactions, but where there are many conflicts between satisfying the different interactions (for instance, an NK landscape with $K \rightarrow N$). This would result in a rugged fitness landscape that is difficult for any method to solve, since there is no structure to exploit. Introducing modularity into this system such that constraints are consolidated may provide a route for some algorithms to provide more efficient optimisation via partially solving sub-components in isolation.

These two contrasting scenarios highlight the importance of the assumed initial reference point before introducing modularity. Either the change in organisation will render a system optimisable by some method, or it will increase the specificity of mechanism that is required in order to optimise.[2] The contrast is closely related to the two perspectives that Wagner (1996) describes in addressing the origin of modularity. First is the 'integration' of sub-components within a system that initially consisted of disconnected parts. Second is the concept of 'parcellation', where initially the system is irreducibly complex and the evolution of modularity introduces leads to a consolidation of constraints. Although components in the unorganised VSM system are not disconnected (as mentioned), integration aptly describes the introduction of modularity into our problem.

[2]Many thanks to Molly Rorick and Devin Drown for our discussions that clarified these concepts.

3.6 Other Test Problems Used In This Thesis

Throughout this thesis, we use a number of problems to test the proposed algorithms. Each of the problems is selected to reveal a facet of the particular algorithm that we are investigating, and hence we do not use the VSM problem for all of the investigations. In this section, we briefly describe and contrast the properties of each of the test problems used. Each problem is fully defined in the chapter where they are first used.

All of the problems that we use have a randomised epistatic to physical linkage map. Therefore, no algorithms can exploit any information provided by physical linkage. Instead, the epistatic linkage must be discovered automatically if it is to be exploited. The one exception to this condition is in Section 4.1, where we investigate the abilities of a genetic algorithm that uses linkage-preserving crossover. In this instance, we do allow the GA to use tight physical linkage assumptions.

We use the VSM for the two studies described in Chapter 4, and in Chapter 7 to test an implementation of MACRO. In Section 4.1, we use the VSM to illustrate a tuneable distinction between a GA with crossover and a mutation-only process. We use a variable strength of structural modularity to provide this tuneable distinction. In Section 4.2 we introduce a symbiosis-inspired algorithm, and use the VSM to demonstrate the widened applicability of this algorithm over previous methods. In Chapter 7, we use the VSM problem to investigate different modes of learning problem structures, and the sensitivity of a MACRO implementation, plus control algorithms, to the weight ratio.

The scalable building-block problem (SBB, Watson and Jansen, 2007) is constructed from several large modules, each of which has two optima. One optima is fitter than the other, and both have equal sized basins of attraction. Several of these modules are concatenated together without interdependencies to create the full problem. The size of the modules, k, and the number of modules, Z, both scale with the problem size. The SBB has 2^Z local optima, only one of which is globally optimal. This global optimum is the configuration with all modules solved to the fitter optima.

The SBB problem is similar to the VSM problem, in that there are an exponential number of local optima and very few global optima, and the local optima are a minimum of k bits apart. However, the reason that we use the SBB in Chapters 5 and 6 is that it does not have any interdependencies between modules. This idealisation means that the highest fitness solution to each module is independent of the configurations of the other modules. Whereas in the VSM the weak between-module dependencies provide a global signal towards the fittest overall configurations, in the SBB, no such signal exists. This lack of information prevents local search methods such as simulated annealing from being able to solve the problem efficiently. Watson and Jansen (2007) proved that certain types of GA can solve this problem in polynomial time, but note that the result depends on a tight linkage map. Overall, this problem exhibits significant difficulty to many classes of evolutionary algorithm. Thus, an approach that is able to automatically identify the structure and subsequently exploit that structure to solve the problem in an efficient manner will in principle qualitatively outperform many alternative evolutionary techniques.

The hierarchical if-and-only-if problem (HIFF, Watson et al., 1998) is a test problem that is built from a single subfunction of if-and-only-if, the Boolean function of equality. Each pair of variables is grouped into pairs using this same subfunction recursively, which results in an overall function that is dependent on all of its variables. The IFF subfunction incorporates epistasis between functional groups starting from the bottom hierarchical layer upward and this causes the optima at each hierarchical level to be maximally distant in Hamming space, creating order-N dependencies at the highest level. This means that HIFF is pathologically difficult for a mutation-only hill climbing process to solve. Nevertheless, HIFF is an ideal candidate for testing a recursive approach that is able to automatically identify problem structure, since the dependencies have a clear modular structure that can be exploited. We use HIFF in Chapter 5 to validate our recursive method.

The straightforward and clean structures in both HIFF and the SBB problems allow us to formally characterise the expected running time of our MACRO algorithm, given in Chapter 5.

In Chapter 6, we modify the SBB by introducing neutrality to the fitter optima such that there are several configurations in each module that are optimal. We call the resultant landscape the scalable building blocks problem with neutrality (SBB-N). This problem allows us to investigate how to handle problem decomposition in cases where the information available from local optima exhibits some ambiguity as to the compatibility of variable values. For a particular module solution in the VSM, SBB, and HIFF problems, the variable values that make up that solution are unambiguous. This leads to a clear modular structure, with no overlap in suitable module boundaries. The clear and unambiguous module solutions are well suited to creating higher-level units (although the discovery of module membership is a non-trivial task, as described above). Conversely, the neutrality in each module in the SBB-N means that there is some ambiguity as to the exact configuration that should make up a higher-level unit.

We also use the problem of finding the ground state of a random bond Ising model (RBIM, Middleton, 1995) in Chapter 6. This problem is defined on a two-dimensional lattice, such that each variable has nonzero interactions with four neighbours. The RBIM has two global optima, when all variables have agreeing values. In these optimal configurations, all dependencies are satisfied (*i.e.*, the RBIM is a consistent problem). However, the problem also exhibits many local optima, which occur when clusters of variables agree locally (a domain), but disagree with neighbouring domains. The dependencies across the domain boundaries will be frustrated, but each single variable change that satisfies one of these dependencies also breaks other dependencies. Hence the configuration is locally optimal.

The domain boundaries at local optima can intersect any subset of variables. Consequently, any pair of variables will not constantly agree in a representative sample of local optima. So although a pair of variable values may agree more often than not in local optima, the correlations are somewhat ambiguous. Moreover, the fact that domain boundaries can occur in many places means that there are many legitimate decompositions. Although this may seem to make the task of identifying a decomposition easier, an automatic method is in fact presented with a more difficult task. The signals that indicate a suitable decomposition are diluted, even when locally optimal configurations are considered (as in MACRO).

The SBB-N and RBIM problems both exhibit some level of ambiguity in which variable values should be joined together: information from multiple local optima will not always indicate the same set of co-occurrences. Moreover, in the RBIM the domain boundaries, which could be used to indicate a suitable decomposition, do not constantly occur at the same positions. Thus, we use these two problems to better understand the conditions under which a probabilistic association mechanism is more appropriate than a mechanism that only forms extreme, discrete associations.

In Chapter 7, we consider a further spin-glass based problem, which we refer to as the rewired random-block problem (RRB). These problems are constructed in a similar manner to the VSM problem, using a pairwise matrix of interdependencies. However, the interdependencies are drawn randomly from $\{-1, +1\}$ for positions on the block diagonal, and all other weights are set to zero. This weight configuration corresponds to the non-rewired case, which provides Z separable blocks. The positions of the nonzero weights are partially or fully randomised, in a similar manner to how the VSM is reorganised for $\rho < 1$. For the fully rewired case, the nonzero weights have no systematic structure in their distribution, and simply correspond to random sparse matrices.

Unlike all of the previous problem classes, the RRB problems are inconsistent in general. That is, no configuration exists that satisfies all of the dependencies. Sub-solutions that resolve many dependencies within a subset of variables can conflict with all of the locally optimal solutions for other subsets. Compare this to, for instance, the VSM. While there are two solutions to each module, and which is fitter depends on the rest of the problem configuration, it is nevertheless the case that one of the solutions can feature in a configuration that satisfies all dependencies (*i.e.*, a global optimum). No such guarantees exist in the RRB problem. Thus, two factors cause particular difficulty in the RRB problem: the inconsistency, and the absence of systematic structure to exploit.

3.7 Conclusion

The VSM construction can create modular functions which are intuitive to understand; the tuneable aspect reveals one dimension in which modularity breaks down, and in this dimension, the difficulty for local search is positively correlated to the strength of modularity. At one extreme parameterisation of weight ratios, the problem can be exponentially difficult for local search, whereas at the opposite extreme there is sufficient information for simple gradient-following algorithms to reliably find a global optimum. A similar contrast can be seen with extreme settings for the structural modularity parameter: modules are present and create ruggedness when the strong dependencies are clustered together to act in concert; and the landscape is trivial when the strong dependencies are sparsely scattered. Nevertheless, in the extremes in both of these axes that defeat local search, the systematic structure present should be exploitable by an appropriate mechanism. In several of the subsequent chapters, we explore mechanisms which can do just that.

In conclusion, the VSM problem addresses our research question. Using a pairwise matrix of epistatic interactions can give rise to a simple problem construction that can exhibit exponential difficulty for local search. Nevertheless, the problem structure can, in principle, be exploited even when the modularity is as strong as possible.

Chapter 4

Search at Multiple Scales, and Problem Decomposition by Symbiogenesis

This chapter describes research that led to our recognition that combining the following components can provide efficient problem decomposition:

- Search at multiple scales, the higher-scale search is guided by the lower-scale search *demonstrated using a genetic algorithm with crossover and mutation variation operators*, (Mills and Watson, 2007b)
- Recursive symbiotic association learning for problem decomposition *demonstrated with the RSSA*, (Mills and Watson, 2007a)

The contributions that are most relevant for our argument are described here, while for fuller documentation of the experiments the reader should refer to the articles cited.

The two papers that this chapter summarises provide support for the utility of the VSM problem (Chapter 3). Mills and Watson (2007b) illustrate the value of being able to vary the strength of structural modularity present in an instance. This paper does so by demonstrating a principled distinction between linkage-preserving crossover (which can exploit modularity) and mutation (which cannot exploit modularity) when the structural modularity is as strong as possible. Conversely, when the modular structures are blurred, the local optima present in the strongly modular case reduce in significance. This dictates a reduction in the degree of discrimination between crossover and mutation. Mills and Watson (2007a) use the VSM to demonstrate a generalisation of the class of problems that symbiosis-inspired algorithms can solve to include non-hierarchically structured problems that use black box optimisation assumptions.

4.1 Multiple Scales of Search: Crossover and Mutation

Crossover mechanisms that preserve linkage have the capacity to keep all of the variable allocations in a module together when creating an offspring: they can provide modular variation. Our aim in this chapter is to better understand where this modular variation can provide an algorithm with a principled distinction in problem solving ability over an algorithm that does not have such a capability. To do so, we investigate the ability of algorithms with and without linkage-preserving crossover in nearly decomposable landscapes. We find that when used in combination with a fast acting mechanism (point mutation), crossover can explore in the space of modules to exploit higher-scale structure in certain landscapes, specifically demonstrated with the VSM (a problem with two scales of structure present). We also find that varying the structuring of the VSM problem leads to a variable level of discrimination between these two classes of algorithm. This finding indicates that the dimension of structural modularity, parameterised in the VSM, controls features that create modular difficulty for search methods that cannot exploit the modular structure (*e.g.*, mutation-only search).

We use a steady-state genetic algorithm (Mitchell, 1996), with deterministic crowding to provide diversity maintenance (Mahfoud, 1992). After selection, variation is applied, which includes both mutation and crossover. Mutation is implemented with a low per-locus probability of a new random allele being introduced, and one-point crossover is applied to a proportion of the offspring. The most informative comparison is with a random mutation hill climber; further controls are also described in (Mills and Watson, 2007b). We test these algorithms on 400-bit instances of the VSM problem (20 modules each of 20 bits), across the full range of structural modularity settings (see Chapter 3). We measure the rate of success of finding a global optimum.

We find that when the modularity is as strong as possible, only the GA with linkage-preserving crossover can find the global optimum; the mutation-only approach is prevented from doing so on account of the ruggedness in the fitness landscape (it finds local optima in every run). At the other end of the scale, where the structural modularity parameter is set to zero, there are no local optima and all algorithms reliably find the global optimum. Moving from no modularity to strong modularity, the mutation-only algorithms degrade gracefully, and only the GA with linkage-preserving crossover remains able to solve the problem in all instances.

The success of the GA can be explained as follows. Mutation finds module solutions, and with a reasonably sized population, it is likely that both solutions to each of the modules are found by some individual in the population, although no one individual is likely to have found all compatible module solutions. Crossover has the ability to combine these module solutions found by mutation, and the inter-module dependencies provide a clear selective pressure towards increasing the number of compatible module solutions to a global optimum.

Above, we attribute the success of using crossover in the GA to the combination of crossover and mutation – because each variation mechanism is efficient at searching a different scale. However, there is an untested logical possibility, that all of the useful search is performed by the crossover mechanism. Here, we describe why both mechanisms do add variation that is essential to the overall process.[1]

As a precursor, we suggest that variation at one scale is not sufficient when a problem has multiple scales of structure. Even if crossover alone could solve the problem reliably, because it performs variation at multiple scales, this does not falsify the claim that multiple scales are required.

Consider a crossover mechanism that was restricted to between-module crossover points, as the sole variation mechanism. This would only be capable of search in combinations of modules, and the module configurations available would come from the initial population (Watson, 2005). Thus, it would require all module solutions to be present in the initial population. This dictates an exponentially sized population with respect to the module size. Conversely, if used in combination with a point-mutation mechanism, the mutation would find module solutions, and the between-module-restricted crossover mechanism could trivially find global optima in the VSM, guided by the weak between-module fitness contributions.[2] As stated (Section 3.2.3), a single-point mutation mechanism cannot escape a local optimum. Therefore, it would only find a global optimum by sampling all of the basins of attraction. This could be achieved either with an exponentially large number of restarts in the number of modules, or equivalently, an exponentially large population size. Therefore, search at either level alone is inefficient, scaling exponentially with some function of the problem size (either $k \propto N$ or $Z \propto N$). On the other hand, combining these two scales of search can efficiently traverse the space to reliably find a global optimum.

So is crossover alone capable of solving the problem? We note that an experiment performed in (Watson, 2005) shows that a one-point crossover operator is able to perform search at both within-module and between-module scales on a two-module problem, but is less efficient than the combination of one-point crossover and point mutation. Elucidating the exact efficiency of crossover alone for problems such as the VSM remains as future work.

Now that we have established that search at two scales is required to solve nearly-decomposable problems efficiently, we identify a related but distinct criticism of the control experiments provided: why compare an algorithm that has mechanisms which operate at two scales against an algorithm that only has a single scale of search?

[1]Essential when using this type algorithm, but not necessarily the only way to solve this class of problem.

[2]Similar conditions to those described for the GA used above would apply: on diversity maintenance; and allowing the mutation process sufficient time to discover the module solutions before suffering disruptions from crossover.

To address this, we must first refine our position to the following. To solve nearly-decomposable problems such as the VSM, a) two search scales are required, and b) the large-scale mechanism must provide variation with correlations that correspond to module solutions. As discussed above, the small-scale search is capable of introducing suitable correlations to the population. To reject this second condition, a counter-argument would need to demonstrate that *uncorrelated* multi-locus changes are sufficient to traverse the higher-scale landscape in a problem such as the VSM.

We note that the local optima identified in the VSM (see Section 3.2.3) are technically only local optima for mechanisms that can change fewer than k bits at once (Jones, 1995b). In principle, a large-scale mutation of k bits could modify the right loci to move from one 'local optimum' to another. However, because modules are large (*i.e.*, k scales with N), a mutation-only method will provably take exponential time to move between these configurations in the limit of $w_I : w_E \rightarrow \infty$.

It is straightforward to define a mutation operator that varies exactly k bits per application, without any correlation in the changes at different loci. An alternative is macromutation (Jones, 1995a), which uses physical linkage information to introduce a new random configuration to one region of the genome. Let us generously restrict the macromutation to vary exactly k bits. Neither of these mechanisms would preserve a module configuration, or be able to (reliably) move from one module solution to another. The former would take $\mathcal{O}\left(2^k \cdot \binom{N}{k}\right)$ trials to do so. The latter mechanism improves matters by exploiting the physical linkage information, but still takes $\mathcal{O}\left(N \cdot 2^k\right)$ trials to move between module solutions.

Thus, while it might technically be fairer to compare crossover and mutation with mutation and macromutation, the computational difficulty outlined above shows that such an algorithm would not yield efficient problem solving. It might be of some interest to identify at what point on the structural modularity scale an algorithm that uses multiple mutation rates would fail, with reference to a single mutation rate algorithm. However, scalable performance cannot be achieved without correlations in the high-scale variation mechanism.

In this section, we have demonstrated that linkage-preserving crossover and mutation are capable of solving a nearly-decomposable problem that mutation alone is incapable of solving in less than exponential time. We have argued that search at either scale alone is ineffective, and hence the combination of variation mechanisms is successful because it provides appropriate multi-scale search.

There are three conditions for this result to hold. First, for linkage-preserving crossover to offer modular variation, the physical to epistatic linkage map must be tight. Second, a diversity maintenance mechanism must be used to prevent convergence on higher fitness local optima that might not contain all of the genetic material necessary for either global optimum. Third, there should be sufficient time for mutation to find all of the module solutions before any are lost through population convergence. We have not directly tested the third stipulation, but its rationale is that if search at the higher scale is to have any useful correlations, the low-scale variation must have had time to act. This provides a possible explanation for the need to use a low probability for crossover events. In Section 8.2 we describe an experiment to confirm this, which uses an an explicit two-stage model that refrains from applying any crossover until no further genetic change occurs under mutation.

Of these conditions, the strongest assumption is that of tight linkage. Ultimately, multi-locus variation with *specific* grouping is best facilitated using an adaptive mechanism. The rest of this thesis develops mechanisms to achieve such adaptive groupings.[3]

4.2 Symbiosis-Inspired Decomposition

In the context of complex problem solving, identifying epistatic structure automatically, and subsequently exploiting that knowledge is one way that allows us to find high-quality solutions. Our focus in this chapter is to consider modular structures in particular. To exploit such modularity, we must vary an entire sub-component at once, rather than varying the individual variables that make up the sub-component independently. Such a mechanism requires the knowledge of module membership – and this organisational information is not likely to be available in general.

There may be some special cases where the epistatic organisation is known, and thus identifying a decomposition automatically is not an issue (*e.g.*, Potter, 1997; Kauffman and Johnsen, 1992). Alternatively, in the case where locus ordering is known, the challenge of locating module boundaries remains. A mechanism such as linkage-preserving crossover may have the potential to be advantageous in this situation, as we saw in Section 4.1. But we are interested in methods that do not depend on these types of restricting assumptions.

Therefore, we propose two research questions for this chapter: Can structures in a modular problem be revealed using a symbiosis-inspired algorithm? If so, can the resulting algorithm efficiently solve problems that exhibit modular interdependency?

To address these questions, we developed an algorithm that utilises the concept of symbiotic encapsulation to create composite units that reflect the problem structure (Mills and Watson, 2007a). The algorithm, the RSSA, identifies pairs of species that are synergistic when they appear together, and joins those species into composite entities that become new units of variation. The process recursively creates coalitions of increasing size and co-dependence to decompose a problem. We find that the RSSA is capable of solving a broader class of modular problems than some prior symbiosis-inspired methods.

[3]Several attempts to adaptively specify crossover points exist in the GA literature. See, for example, (Li and Goodman, 2007; Harik, 1999).

Algorithm

The RSSA deviates from a more traditional evolutionary algorithm in several features. Entities have a variable length genotype, and in general are partial specifications. That is, an entity only specifies values for a subset of the loci in the problem. To make this approach compatible with black-box function evaluation, entities are evaluated in a context, which defines values for all loci that are not specified by the entity being tested. The entities reside in an 'ecosystem', and each represents distinct functionality rather than presenting a multitude of (fully specified) variants for the same task. The only variation operator is a symbiogenic join, which takes two entities and returns their superposition. If both entities specify the same locus, the first entity considered takes precedence.

The ecosystem evolves in two phases. The first phase evaluates the synergy (see Equation 4.2) between all pairs of entities currently in the ecosystem within a number of randomly generated contexts. The second phase selects a number of pairs of entities to be joined, each of which reduces the size of the ecosystem, E, by one entity (as it has been encapsulated to create a larger composite). Joins are made between pairs that reveal maximal *reciprocal* synergy: both partners must 'want' a join for it to be made. We refer to one complete cycle through these two phases as an 'epoch'.

Table 4.1 provides pseudocode for the RSSA. C is the number of contexts each proposed join is tested within, N is the system size (which implies $2N$ primitives that make up the initial atomic entities), and J is the number of joins made per epoch. J can be interpreted as a form of control on the conservatism of the joining.

We define a fitness delta, δ, as the difference in fitness, $f(\cdot)$, between a context, c, alone and an entity, A, in that context (Equation 4.1). Synergy, s, is defined as the difference in fitness deltas of the symbiont, $A+B$, and the sum of the individual entities (Equation 4.2).

$$\delta(A,c) = f(A+c) - f(c). \tag{4.1}$$

$$s(A,B,c) = \delta(A+B,c) - \delta(A,c) - \delta(B,c). \tag{4.2}$$

This measure aims to quantify the additional benefit obtained by coexisting, over individual existence. This is closely related to what might be called a measure of epistasis between A and B when in the context of c. The selection phase is a competition between potential joins, and the most viable are those with high synergy scores. The symbiotic relationships are created such that whilst high synergy is rewarded, both symbionts must be 'interested' in the join being made. We find this *reciprocal synergy maximisation* to be key to assigning the correct symbiotic joins.

Table 4.1: Pseudocode for the Reciprocal Synergy Symbiosis Algorithm

```
1.  Initialise ecosystem with one copy of each primitive entity in the substrate
2.  While exit conditions not met (some remaining entities have length $< N$)
      a. For $i$=1:$C$
          i. generate context $c$ as random configuration
          ii. evaluate context $c$, $f(c)$
          iii. For each ordered pair of entities remaining in ecosystem, $A, B$
              (1) Evaluate $A$ in context $c$, $f(A + c)$
              (2) Evaluate $B$ in context $c$, $f(B + c)$
              (3) Superimpose $A$ and $B$ ($\rightarrow A$+$B$)
              (4) Evaluate $A$+$B$ in context $c$, $f(A + B + c)$
              (5) Calculate synergy for $A$+$B$, according to Equation 4.2
      b. For each ordered pair of entities $A, B$
          i. calculate mean synergy values across all $C$ contexts
      c. Process synergy matrix:
          i. Rescale mean synergy values to range $[0, 1]$
          ii. Calculate reciprocal synergy for each pair: $s(A, B) \times s(B, A)$
      d. Permanently join $J$ pairs with the highest reciprocal synergy values: $A \leftarrow$
    $A$+$B$, and remove $B$ from the ecosystem
3.  Report final ecosystem
```

Results

We test the RSSA on two modular problems: 1) HIFF, with a hierarchical structure (Watson et al., 1998), and 2) the VSM, with a flat modular structure (Chapter 3). We find that our new algorithm can solve large instances of both of these modular problems (400-bit VSM, and 256-bit HIFF), and that it does so by discovering the structural modularity and subsequently exploiting it. In the case of HIFF, the reciprocal synergy measure reveals structure from a small number of hierarchical layers. When joins are made to reflect this structure, the search is scaled up and further layers of hierarchy are revealed.

We demonstrate that the RSSA and a genetic algorithm with linkage-preserving crossover (as detailed in Section 4.1) are capable of solving the VSM problem only under tight linkage assumptions. When linkage information is removed (by randomising the genetic to epistatic linkage map), the RSSA is unaffected, but the GA can no longer solve the problem. We also compare with SEAM (see Section 2.5), and find that it is unable to solve VSM problems.

For the VSM experiments, we set N=400, Z=k=20, wI=400, wE=1. The RSSA uses C=160 contexts, and $J = \lfloor 0.5E \rfloor$ joins are made in each epoch, where E is the number of entities remaining in the ecosystem. In 30 repeats, a global optimum is found in a mean of 439,574,532 evaluations (s.e.i.m. 13,732) for the shuffled VSM problem, and 439,599,949 evaluations (s.e.i.m. 15,998) for the unshuffled VSM problem.

For the HIFF experiments, we set N=256, giving L=8 hierarchical layers. The RSSA uses C=160 contexts, and $J = \lfloor 0.5E \rfloor$ joins are made in each epoch. In 30 repeats, a global optimum is found in a mean of 166,999,650 evaluations (s.e.i.m. 30,010) for the shuffled VSM problem, and 167,007,374 evaluations (s.e.i.m 30,085) for the unshuffled VSM problem.

Discussion

Simplifications Over SEAM. We highlight three aspects in which the RSSA and SEAM differ: 1) the joining mechanism, 2) the parallelisation, and 3) the method for context generation.

The Pareto dominance mechanism that SEAM uses is very conservative, demanding a fairly exhaustive verification that no better symbiotic partner exists. We replace this mechanism in the RSSA with a simpler mechanism based on a measure of synergy, which more directly assesses and exploits the epistasis in the problem. By making a number of joins in each epoch, according to pairings with the strongest synergy, the state of system decomposition is advanced continually.

In order to allow a competition for joins based on synergy values, the RSSA moves to implement in a batch mode. SEAM operates under a steady state mode, testing one proposed join at once. While the steady state mode may be more biologically plausible, we find that the batch mode calculations offer two advantages computationally. First, by separating out the two phases of 1) gathering information, and 2) structural decomposition, it makes our aims of scaling up search, a layer at a time, more explicit, particularly when operating on a hierarchical problem. More importantly, the synergy measure reveals the model of the structure that has evolved. This information is straightforward to interpret, and increases our confidence that the algorithm operates successfully because it correctly decomposed the problem.

Where SEAM required coevolved contexts, the RSSA shows that randomly generated templates are sufficient to facilitate the evaluation of partially specified entities.

On Improving Efficiency. The algorithm has two main components that dictate the number of fitness evaluations that are used: the synergy calculations, and the number of contexts that each candidate join is tested in. The synergy calculations use three fitness evaluations per pair of entities in each context, and additionally the context alone must be evaluated. Therefore, each epoch uses $\left(3 \cdot (2N)^2 + 1\right) \cdot C = \mathcal{O}\left(N^2 \cdot C\right)$ evaluations.

We have not derived the expected number of contexts that will provide a sufficient sample to make appropriate joins, nor the relationship between sample size and the system size. These characterisations are left for future work. Nevertheless, understanding how to reduce the number of contexts used per decision is one area that we could improve upon to solve the same class of problems more efficiently.

Many contexts are needed to prevent spurious joins between non-interacting variables. However, significantly fewer contexts already provide a good estimate of the structure. In some preliminary investigations we focused the computational effort on the subset of joins that show promise initially, thereby allowing further effort to be spent increasing confidence on the suitability of proposed joins. Under this focussing regime, if after s contexts, the top half of remaining candidate joins are considered for a further s contexts, the total number used for equivalent accuracy will be $C' = \mathcal{O}\left(\frac{C}{\log_2 N}\right)$.

This scheme reduces the mean number of contexts used per join, and may improve the efficiency of the algorithm. However, this or similar schemes do not address a more fundamental issue that may improve the applicability.

Random Contexts are Noisy. With hindsight, we acknowledge that part of the difficulties that the RSSA has in identifying the correct joins stem from the linkage detection attempts from noisy contexts. Both the RSSA and SEAM use fairly complicated mechanisms to identify epistatic information from a large set of contexts. Without any guidance as to which pairs to consider for joining, it is not obvious on how to simplify those mechanisms in a manner that will provide a qualitative difference (in applicability, or cost of solving the classes of problem tested here). From a seemingly unrelated angle, we can also now observe that the RSSA does not exploit the current units before trying to adapt those units. That is, the algorithm spends all of its effort assessing which new units to create. These two points feed into the developments in the next chapters, where we develop mechanisms that do not directly use fitness evaluations to facilitate each decision between every pair of species. Instead, we pool resources by expending fitness evaluations on reaching an ensemble of local optima, and make symbiotic joins between entities that co-occur frequently across local optima. We subsequently develop these principles in variant algorithms that can gradually increase associations such that the variation distribution is modified without needing to make a permanent and extreme join.

In summary, we have developed an algorithm that is simplified over prior symbiosis-inspired algorithms. Although comprehensively describing the algorithmic niche of the RSSA remains an open question, we are confident that our simplifications allow the RSSA to have a more general application than SEAM. The only problem SEAM is shown to be competent on is shuffled-HIFF whereas the RSSA is here demonstrated to solve both shuffled-HIFF and another, less contrived, form of modular problem that SEAM cannot solve. The application of structural decomposition in a batch mode reveals the model of the structure that has evolved, which accurately reflects the structure in the problems tested.

Chapter 5

A Framework for Symbiotic Optimisation

In all but the most trivial of optimisation problems, there are dependencies between the most appropriate allele at one locus and the alleles that are present at other loci: the optimal setting is context-dependent ('sign epistasis', Weinreich et al., 2005). In order to resolve those dependencies, the approach that we take is to evolve relationships between species (initially single alleles) such that those species co-vary, effectively creating larger entities that now specify values for multiple loci. Provided that good decisions can be made regarding joining species that are compatible with one another, a solution can be constructed by composing larger and larger sets of species that are harmonious.

We highlight two key findings from the previous chapter that are influential in the new material in this chapter. First, in Chapter 4 (a) we demonstrated how using two variation mechanisms at different scales are effective at solving problems with modular interdependency: mutation provides a small scale of variation that finds module solutions; and crossover provides a larger scale of variation that can search combinations of module solutions (provided that the problem has tight linkage that crossover can exploit). Second, in Chapter 4 (b) we demonstrated that symbiogenesis can provide a method to mix module solutions to explore the higher level search space, doing so independently of physical linkage information.

Given these findings, and the aim of creating more efficient and reliable problem decomposition techniques, we address the following question in this chapter:

> How is it possible to identify and exploit the appropriate symbioses such that the scale of evolution suits the structure in the landscape?

The advances made in this chapter lead to a simpler algorithm that is more efficient in its problem-solving ability than prior symbiosis-inspired algorithms. We achieve this by

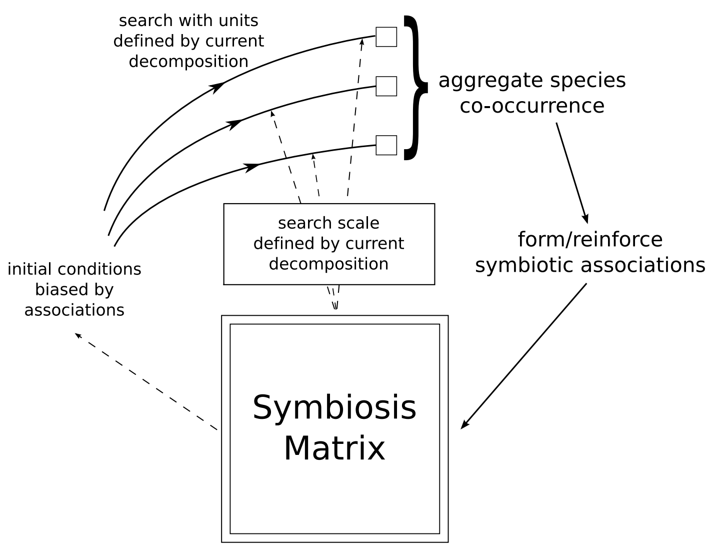

search with units defined by current decomposition
aggregate species co-occurrence
search scale defined by current decomposition
form/reinforce symbiotic associations
initial conditions biased by associations
Symbiosis Matrix

Figure 5.1: Schematic of the MACRO algorithmic framework. Symbiotic associations are formed between species that co-occur frequently in locally optimal configurations, and stored in the Symbiosis matrix. This matrix prescribes both the variational steps and the initial conditions for search in later epochs. Specifically, once symbiotic associations have been formed, subsequent search uses correlated multi-locus variation and thus occurs at the macro-scale.

separating the timescales upon which existing symbiotic relationships are exploited from the timescale where new symbioses are created. Specifically, symbioses that currently exist are exploited on a rapid timescale, and new symbioses are created on a slower timescale. Thus, symbioses are modified according to information from locally optimal configurations. These are significantly enriched over the random contexts used in the RSSA, and consequently far fewer contexts are required to make appropriate joining decisions.

The developed algorithm performs hill-climbing in several independent demes, each deme finding different local optima. When the co-occurrence of species in the ensemble of local optima is as strong as possible, symbiogenic joins are formed between those sets of species. The joins reflect the structure of the problem, and enable macro-scale variation in subsequent search. In the original hill-climbing, only single-locus changes could occur, but once symbiotic joins are formed, each step of hill-climbing can involve a specific multi-locus change. The process recurses to repeatedly find and exploit higher-order decompositions. See Figure 5.1.

In summary then, the main contributions of this chapter are as follows.

This chapter demonstrates that our proposed algorithm can efficiently solve structured search problems.

We perform experiments on test problems that exhibit modular interdependency. The modularity in the problems tested creates significant ruggedness, and this defeats large classes of weak stochastic search algorithms. Nevertheless, we demonstrate empirically and support analytically a log-linear time complexity for our proposed algorithm on these problems. Furthermore, we show that our algorithm uses local optima discovered by micro-scale search to guide the formation of symbiotic associations that reflect the problem structure. This decomposition, which enables macro-scale search, is responsible for the problem solving efficiency of our algorithm.

5.1 Compositional Algorithms in Evolutionary Computation

Three key concepts lie at the foundation of the algorithms that we propose: probabilistic model building, macro-scale hill-climbing, and recursion. Model building and recursion are both frequently used in the stochastic optimisation literature, as are local search techniques. Methods that combine two of these three concepts are not uncommon. However, *macro-scale hill-climbing* is much less common. Here we review relevant literature, and conclude that there are few examples of using macro-scale hill-climbing in combination with recursion (and model building) in a single method in the prior literature. These papers are described below.

A number of algorithms that take inspiration from symbiosis and symbiogenesis have been suggested in the past. The paradigm that these algorithms aim to exploit is compositionality (Watson, 2006): where pre-adapted components (sub-solutions) are combined via symbiotic encapsulation, to provide larger sub-solutions and ultimately lead to high-fitness fully specified solutions. The most explicitly compositional algorithms include SEAM (Watson and Pollack, 2003), ETA (Defaweux et al., 2005) and the RSSA (Mills and Watson, 2007a). All of these works describe some conditions under which they can use symbiotic relationships to automatically decompose problems.

The formation of symbiotic relationships in these algorithms leads to the construction of new composite variables that group together several pre-existing variables. Over evolutionary time, composites are recursively formed to specify a larger subset of the problem space. Each time this occurs, variables that were previously freely able to vary independently have become a single unit, and can only co-vary: the dimensionality of the search space is thereby reduced.

We can therefore identify two key concepts in symbiosis-inspired algorithms. First, *association formation*, which controls the nature of subsequent variation that occurs in the system, specifically by enabling large, but non-random genetic changes. Second, *recursion*, which arises from the repeated application of symbiotic encapsulation of larger and larger components that describe compatible functionality of an increasing portion of the system.

The main novel concept introduced in this chapter is macro-scale hill-climbing. In this chapter we apply many variation steps before forming any new associations, such that a locally optimal configuration is discovered. It is important to stress that this process is much more than simply applying a local search operator. The variational steps are defined by the associations that have been formed, and are therefore large-scale (but non-random) changes: we define these association-informed multi-locus changes as *macro-scale* units. Search using macro-scale units is in a configuration space that has lower dimensionality than the original system. Each time more associations are made, the dimensionality is further reduced.

In the remainder of this section, we briefly examine work that is related to the concepts identified above.

5.1.1 Local Search, and Selecting Initial Conditions

The simplest form of stochastic search is the perturbation of an initial solution, possibly random, by single variable changes. The locally optimal configurations that such a process will find have higher average fitness than the set of all configurations. However local search is a two-edged sword: escaping local optima is a central problem for evolution (Wright, 1935) and by extension, local search based on principles of evolution. By definition such an escape requires multi-locus changes, the magnitude of which may mean that this is infeasible by simply turning up the mutation rate.

Local optima are fitter on average than a random configuration because at least some dependencies are satisfied; if any perturbation can increase the number of dependencies satisfied (or overall utility), local search is likely to accept such a perturbation. Except in rare circumstances, landscapes will exhibit multiple local optima and therefore not all of the dependencies will be satisfied at the end of a local search run. However we may be able to glean information from local optima nonetheless.

There are several types of algorithm that aim to use only local optima in order to search a rich subset of the total configuration space – simply restarting from new random initial conditions can gain some traction (Johnson, 1990). Some iterated local search techniques aim to bias the sampling of initial conditions to improve upon *random* restarts (Lourenco et al., 2003). One example of this class is Chained Local Optimisation (Martin and Otto, 1996). This algorithm accepts or rejects new states based on a comparison of the quality of the local optimum at the end of a trajectory, to ensure decisions are based on local optima. A 'kick' operation is applied to move the trajectory to a new basin of attraction (though what makes a successful kick is problem-specific).

Using local search to adjust the balance of exploitation versus exploration is the kernel behind memetic algorithms: selection only occurs on local optima (although the inheritance scheme may transfer initial conditions or final states of the local search, depending on whether a Lamarckian or Baldwinian scheme is applied). See Krasnogor and Smith (2005) for a review of memetic algorithms.

Some algorithms put effort into selecting fruitful initial conditions for local search runs. For instance, STAGE aims to provide initial conditions that will lead to optima of increasing quality, by learning features of the search space through modelling likely outcomes based on previous experiences (Boyan and Moore, 2000). Qasem and Prügel-Bennett (2008, 2009) use clustering of local optima to determine seed configurations for further search. The second search phase can use a different operator from the first phase, and the authors have found this method to be successful on MAX-SAT and vertex cover problems. COMIT uses locally optimal configurations to build a model, which is then sampled to provide initial conditions for subsequent local search runs (Baluja and Davies, 1997a, 1998).

However in most of these cases, the variation steps used in the local search are bit-flip or small perturbation. Chained local optimisation suggests the use of larger-scale perturbations (called kicks) to move about the search space. Crucially, these kicks are not adaptive at all.

5.1.2 Problem Decomposition

One alternative to modifying the initial conditions for subsequent search is to break down a problem into smaller sub-problems and optimising those sub-problems in partial isolation. Cooperative coevolutionary algorithms (CCEAs, see *e.g.*, Potter, 1997) take this approach, employing multiple sub-populations to solve each of the sub-problems. Representative members from each sub-population are combined to give candidate solutions to the full problem. However, CCEAs use a pre-specified problem decomposition, which limits their applicability to problem classes where labelling is available. Several other algorithms take a similar approach, also using *a priori* knowledge of the system structure (*e.g.*, Kim and Kim, 2005; Shakya, 2006; Wallin et al., 2005).

5.1.3 Automatically Detecting Epistatic Structures

Estimation of distribution algorithms (EDAs) build models of the distribution of selected candidates in the current population, and sample that model directly to produce the new generation of candidates. Many different types of model have been used in EDAs, including trees and forests (Baluja and Davies, 1997b; Pelikan and Mühlenbein, 1999), Bayesian networks (Pelikan et al., 1999; Larrañaga et al., 1999), and Markov networks (Santana et al., 2006; Shakya and McCall, 2007). The type of dependencies that can be represented in these models varies, but the aim is essentially the same: to model the structure of epistatic interactions in the problem. Thus, when new candidates are drawn from the modelled distribution, variable assignments are conditioned upon the values of variables that they depend upon, such that the likely subset configurations reflect fitter candidates from previous generations.

The way that COMIT uses its model is an interesting example that lies between EDAs and iterated local search techniques. Hill-climbing is used to find local optima, copies of which replace some of the current population. A model is built from this new population. However, when samples are drawn from the model they are used to define seeds for subsequent hill-climbing runs.

A different angle is taken by some model-building algorithms, where multi-locus variation is applied in order to perturb candidate solutions drawn from the probabilistic model. Unlike a macro-mutation (see Jones, 1995a), multi-locus variation can be informed by the probabilistic model (*e.g.*, Handa, 2007): dependents of a mutated variable are re-sampled according to the model. However, when a single such event is applied, it is not clear what useful variation this offers that was not available simply by drawing candidates from the original probabilistic model. Moreover, the observed improvements from such a perturbation suggest the presence of erroneous associations in the model.

Further, some hybrid EDAs exist where local search is applied to candidates before selection (*e.g.*, Pelikan and Goldberg, 2003; Zhang et al., 2004) such that models are constructed from local optima.

Few EDAs perform their model building recursively, or in any kind of compositional manner. However, the DSMGA++ is an exception (Yu and Goldberg, 2006). This uses mutual information measures to construct a matrix of interdependencies, and then performs a search in the space of possible model allocations (DSMs). Variation is introduced by using 'building-block-wise crossover', which prevents any variable settings within a block (in the DSM model) from being disrupted.

5.1.4 Symbiosis as Model Building

One way to view symbiosis-inspired algorithms is as building (probabilistic) models of problem structures, using the model to provide non-random, multi-locus variation in subsequent search. When sets of variables are reduced to always appear together as symbiotic unions, the search space dimensionality is explicitly reduced. Note that the corresponding model from using an encapsulation operator is deterministic in its associations (that is, they are all-or-nothing joins between species). EDAs aim to create a model of a system that well describes subsystems, and create higher fitness fully-specified solutions through the combination of these sub-solutions. Symbiosis-inspired algorithms may be more explicit about this composition of sub-solutions, but the aim certainly underlies EDAs too. Indeed, in the next chapter, we explore a probabilistic variant of the algorithm that we propose in this chapter, which moves closer to EDAs.

If such a process is applied repeatedly (*i.e.*, unions are formed between multi-variable sets), the search space dimensionality can be broken down recursively, allowing great efficiencies in highly structured environments. Associated sets of variables vary at once, according to the current model. New associations are formed according to the fitness (or some other measure of suitability) of a proposed join.

5.1.5 Prior Symbiotic and Compositional Algorithms

The concept of symbiosis has been the subject of several studies in evolutionary computation and artificial life. Of particular relevance to this thesis is the view that a symbiotic mechanism can combine large and non-random modules from multiple pre-adapted sub-solutions.

Bull et al. (1995) investigate the conditions under which endosymbiosis is favoured over free-living coevolution, using Kauffman's NKC landscape (see also Bull and Fogarty, 1995). This work provides a useful analysis of where stabilising selective contexts is favoured.

SEAM is an early evolutionary algorithm inspired by symbiogenesis (Watson and Pollack, 2003). It differs from typical evolutionary algorithms in several aspects. These include the symbiogenic variation operator, an ecosystem of entities that make partial specifications, and coevolved ecosystem templates that are used to facilitate evaluation. The main pool of genetic material represents an ecosystem of many different entities, each specifying only part of an overall solution. All genetic variation is facilitated via the symbiotic join of two entities. The main loop of the algorithm operates as follows: two entities are picked at random, and evaluated in a number of contexts to determine if the pair should make a permanent symbiogenic alliance. A version of Pareto-dominance (see Watson and Pollack, 2003) is used to make these decisions. If this is the case, the two symbionts are removed from the ecosystem and replaced with the chimera. As this process is repeated, the average size of entities will increase until fully specified solutions are discovered.

ETA (Defaweux et al., 2005; Defaweux, 2006) takes cues from models of multi-level selection (Lenaerts et al., 2001, 2003), as well as the compositionality of a symbiogenic transition from SEAM. ETA uses symbiogenic encapsulation to construct variable setting groups, and is applied to constraint-satisfaction problems (which allows partial evaluation). It takes a coevolutionary approach to ensure compatibility before a 'transition' (composition of variables). The initial entities are individual variable assignments.

The RSSA follows a similar framework to SEAM, with a simpler mechanism to assess joins (Mills and Watson, 2007a, see also Chapter 4). Join decisions are made by testing the synergistic benefit given when a pair of entities is joined across several genetic backgrounds, and selecting the relationships that provide the strongest reciprocal synergy. One feature of the RSSA is that it made some progress with revealing problem structures.

The HGA (de Jong et al., 2005b) is a computational abstraction of SEAM, but uses the concept of context-optimality to make decisions on variable assignments. That is, entities are tested in several backgrounds, and only the configurations that are found to be optimal in at least one genetic background are retained. This is applied recursively to hierarchically decompose problems.

The work of Philemotte and Bersini (2007) suggests using 'lenses' to identify variable subsets that are only considered as a single unit during one phase of evolution (a collection of cities are considered as an unchangeable region in the example of TSP). The lens memberships are subjected to a separate evolutionary search, with lenses that lead to better results being selected and modified. Philemotte and Bersini (2006) use a similar idea with simpler lenses. There is no explicit composition of lenses, however lenses suitable for a hierarchical problem are evolved.

Toussaint (2003) analyses the exploration distributions that mutation, crossover, and EDAs provide, and concludes that neither mutation nor crossover can increase mutual information in their exploration distributions. However, EDA models are capable of generating correlated exploration distributions that have increased mutual information *and* entropy. Toussaint and von Seelen (2007) describe developmental schemes (genotype-phenotype mappings) that correlate the variation in the phenotype via small changes in the genotype leading to many changes in the phenotype. Toussaint suggests that the approach nature takes to evolve solutions is to use "simple adaptation on suitable representations" (Toussaint and von Seelen, 2007, p778). Toussaint (2005) considers a compression scheme based on L-systems for genotype representations that leads to 'compact genetic codes'. These give rise to an efficient exploration of highly structured spaces, by enabling the repeated expression of sub-components. The authors demonstrate the hierarchical structure of HXOR (Watson et al., 1998) is amenable to the compact genetic codes approach. Clune et al. (2008, 2009) also consider evolving indirect encodings (genotype-phenotype mappings) that give rise to a different search neighbourhood, which may also be appropriate for some tasks that exhibit regularity.

5.1.6 Model-Informed Local Search

Local search as described above (Section 5.1.1) is defined over the original system units, and when applied, leads to local optima. Of course, the configurations that are locally optimal depend entirely on the variation type that is in use ("one landscape for one variation operator", Jones, 1995b). Let us consider a bit-flip hill-climber as a canonical local search algorithm. This is easily extended, provided we have a model of what variables should change together. Instead of changing just a single variable, we can change an entire group in each step. It is important to stress that this is not a macro-mutation (many uncorrelated changes): using a model of variable dependencies provides correlations in the variation. This can be qualitatively distinct variation from using macro-mutations, provided that the model represents non-random information. Therefore, model-informed local search can be implemented simply, as can uncorrelated local search, but non-arbitrary multi-locus changes will occur between each evaluation.

In the algorithm proposed in this chapter, the units of variation are adapted as the algorithm runs. Consequently, configurations that were locally optimal for a single-bit-flip hill-climber may not be local optima if suitable modular variation has evolved. Suggestions of algorithms that feature local search, model building, and recursion can be found in a handful of papers (Mahdavi et al., 2003; Houdayer and Martin, 1999; Lima et al., 2006), but its effectiveness was only appreciated recently by (Iclanzan and Dumitrescu, 2007). Houdayer and Martin's algorithm uses information on the lattice in spin glass problems; Mahdavi et al. explore only one phase of model-informed local search; and the work by Lima et al. emphasises using local search within the decomposition found, rather than any manner of search that uses units defined by the decomposition. Iclanzan's algorithm is simpler and is more explicit in identifying the reasons for success. The algorithm presented in this chapter is similar to that of Iclanzan's.

In the next chapter, we develop a generalised algorithm that does not use absolute and irreversible joins, instead allowing probabilistic joins. We show that using explicit dimensional reduction has some limitations that are not suffered by the generalised algorithm. We shall return to a fuller comparison with these works in Section 6.5.1 once we have described and examined our own approach.

5.2 Symbiotic Optimisation Algorithm

The algorithm we present comprises two main stages in its loop. The first stage uses a simple hill-climbing process in several independent demes (sub-populations) to find ecosystem configurations that are optima. In the second stage associations are reinforced between species according to the frequency of their co-occurrence in these local optima. The formation of associations leads to macro-scale units that are used in subsequent hill-climbing at the higher scale. These two stages are repeated. Hill-climbing at the macro-scale leads to the discovery of different 'local' optima, which in turn inform which associations will be further reinforced. Figure 5.2 provides an overview schematic of the algorithm, (a) illustrates the hill-climbing process, while (b)–(d) together lead to macro-scale units.

There are many different implementations that would realise an algorithm under this general framework, using parallel hill-climbing search to inform the formation of associations, and using those associations to inform subsequent rounds of higher-scale hill-climbing. In this chapter we investigate a minimal instantiation, in order to provide intuition as to its operation. Here, associations that can form between species are restricted to binary, all-or-nothing relationships. In the following chapter we investigate a richer instantiation where probabilistic relationships between species are permitted.

We present a series of diagrams to illustrate how the algorithm operates, before providing a description of MACRO algorithm. The figures 5.2–5.11 describe the MACRO framework, and while some details are specific to the algorithm investigated in this chapter, the series illustrates key concepts that are relevant for all three instantiations of the framework in Chapters 5, 6 and 7. To provide a concrete working of our algorithm, we use a 25-bit instance of the VSM problem as introduced in Chapter 3 (and do not shuffle the linkage map in order to aid the visualisation).

As discussed in the previous section, several components used in our algorithm are not novel in themselves. However, we find it appropriate to provide detail on the full process in order to show how the mechanisms fit together, as well as to illustrate our algorithm representation.

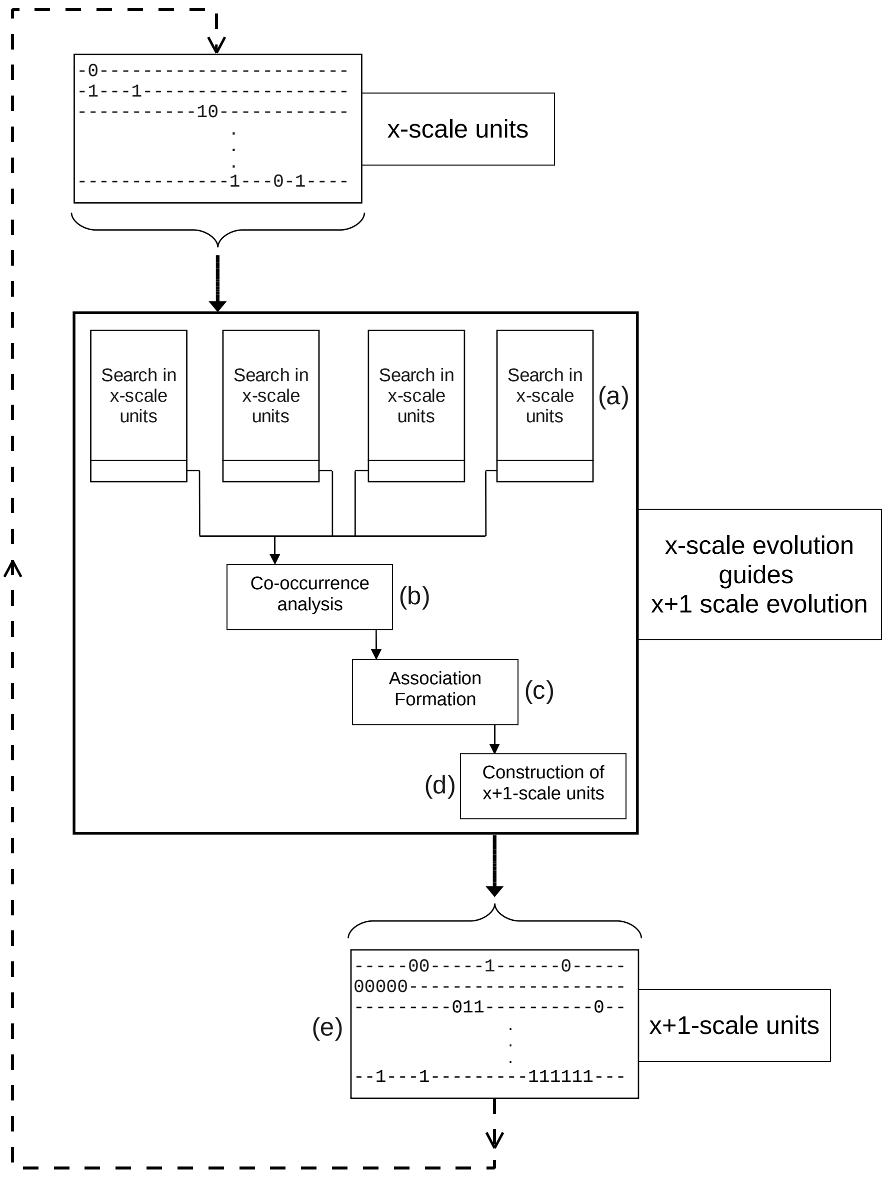
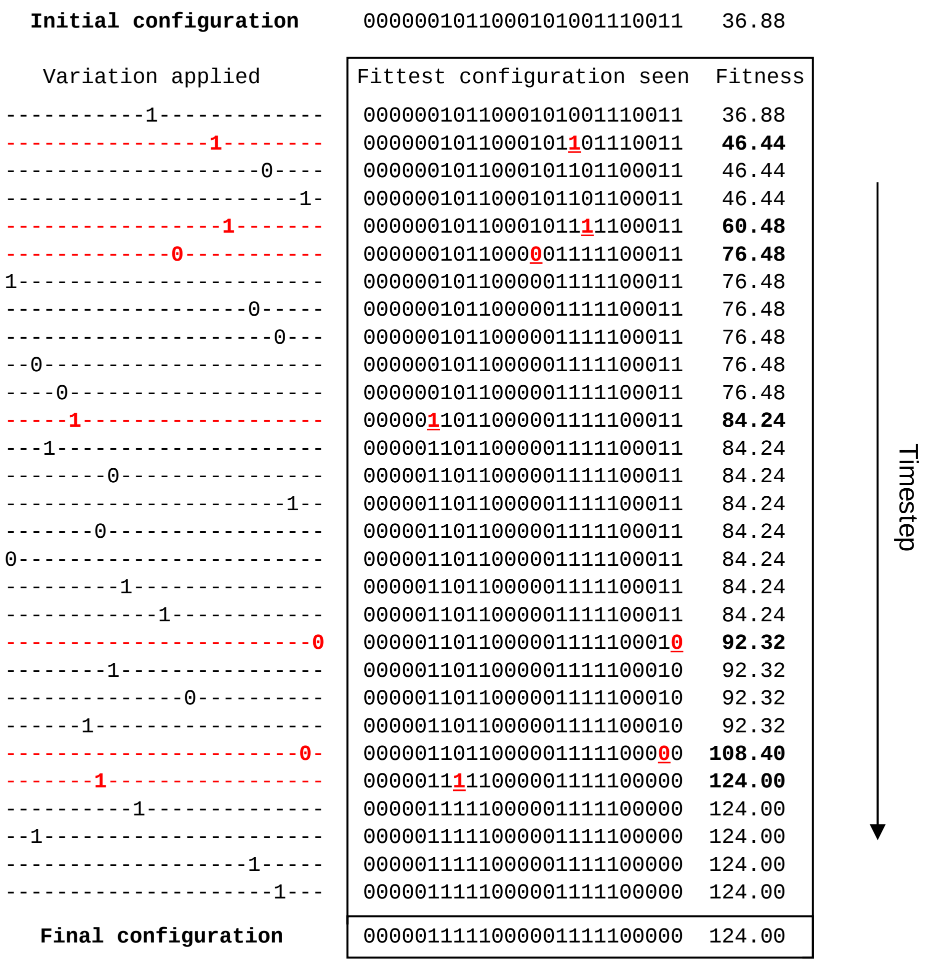


Figure 5.2: The main stages in MACRO: overall, one iteration through all stages scales up the units that are available for subsequent search. (a) Search is performed in parallel in several independent demes, with units at the current scale, x. See Figure 5.4. (b) The set of local optima found in (a) are used to calculate the co-occurrence of x-scale units. Figure 5.6 illustrates the process of calculating co-occurrence, and Figure 5.7 shows an example of the resulting co-occurrence matrix. (c) The co-occurrence data is used to determine which x-scale units will become associated (see Figure 5.8). (d) The symbiotic associations are used to form groups of x+1-scale units from x-scale units, which reflect the structure of interdependencies in the problem (Figure 5.9). (e) This set of x+1-scale groups make up the output of one pass through the algorithm (Figure 5.10). The final figure in the sequence provides an example of search with higher-scale units (5.11).

```
----0----11----0----01--- A
0100100010111010010110100 Current configuration
0100000011111010010101100 New configuration
```

Figure 5.3: An illustration of the superposition of a partially specified unit onto a fully specified configuration. All of the values specified in the unit A are asserted in the new configuration, while remaining values are taken from the current configuration. The variables whose values changed are highlighted in the example. Note that superimposing this unit has not made a change in all variables that it specifies, since some values match those already assigned in the current configuration.



Initial configuration	0000001011000101001110011	36.88
Variation applied	Fittest configuration seen	Fitness
-----------1-------------	0000001011000101001110011	36.88
----------------1--------	0000001011000101101110011	**46.44**
--------------------0----	0000001011000101101100011	46.44
-----------------------1-	0000001011000101101100011	46.44
-----------------1-------	0000001011000101111100011	**60.48**
-------------0-----------	0000001011000001111100011	**76.48**
1------------------------	0000001011000001111100011	76.48
-------------------0-----	0000001011000001111100011	76.48
---------------------0---	0000001011000001111100011	76.48
--0----------------------	0000001011000001111100011	76.48
----0--------------------	0000001011000001111100011	76.48
-----1-------------------	0000011011000001111100011	**84.24**
---1---------------------	0000011011000001111100011	84.24
--------0----------------	0000011011000001111100011	84.24
----------------------1--	0000011011000001111100011	84.24
-------0-----------------	0000011011000001111100011	84.24
0------------------------	0000011011000001111100011	84.24
---------1---------------	0000011011000001111100011	84.24
------------1------------	0000011011000001111100011	84.24
------------------------0	0000011011000001111100010	**92.32**
--------1----------------	0000011011000001111100010	92.32
--------------0----------	0000011011000001111100010	92.32
------1------------------	0000011011000001111100010	92.32
-----------------------0-	0000011011000001111100000	**108.40**
-------1-----------------	0000011111000001111100000	**124.00**
----------1--------------	0000011111000001111100000	124.00
--1----------------------	0000011111000001111100000	124.00
-------------------1-----	0000011111000001111100000	124.00
---------------------1---	0000011111000001111100000	124.00
Final configuration	0000011111000001111100000	124.00

Figure 5.4: Initial hill climbing, using the atomic units of the system. The example problem is a 25-bit VSM problem with 5 modules, w_E=1/100. Each of the 50 atomic units specifies a value for a single variable in the problem. After initialising to a random configuration, several units are superimposed sequentially (see Figure 5.3), and if the fitness of the overall configuration is non-worse the change is retained. The highlighted variations show cases where the configuration was improved and the variation therefore retained. This process will find a local optimum – once none of the current units can improve the configuration.

```
 1 0000011111000001111100000
 2 0000000000111110000000000
 3 1111100000000001111100000
 4 1111100000000001111111111
 5 0000011111111111111111111
 6 1111111111000000000011111
 7 0000011111000000000011111
 8 0000000000000000000011111
 9 1111111111111110000011111
10 0000011111000000000000000
11 1111111111111110000011111
12 1111111111111111111100000
13 0000011111111111111111111
14 0000000000000000000011111
15 1111111111111110000000000
```

Figure 5.5: An ensemble of 15 locally optimal configurations, discovered by multiple runs of the process in Figure 5.4. None of the runs found a global optimum (which in the VSM are at all-0 and all-1). The local optima are nevertheless an information rich subset.

0------------------------ A -0----------------------- B	A,B co-occur in: 1,2,5,7,8,10,13,14.	U(A)=8/15, U(B)=8/15 CO(A,B)=8/15
----1-------------------- C -----1------------------- D	C,D co-occur in: 6,9,11,12,15.	U(C)=7/15, U(D)=10/15 CO(C,D)=5/15
----------------0-------- E -----------------------1- F	E,F co-occur in: 6,7,8,9,11,14.	U(E)=9/15, U(F)=9/15 CO(E,F)=6/15

Figure 5.6: Example co-occurrence calculations for atomic units present in the set of final deme configurations of Figure 5.5. U(X) denotes the univariate occurrence of unit X (*i.e.*, the number of configurations in which X is present). CO(X,Y) denotes the co-occurrence of units X and Y (*i.e.*, the number of configurations in which both X and Y are present).

	0------------------------	-0-----------------------	--0----------------------	---0---------------------	----0--------------------	-----0-------------------	------0------------------	-------0-----------------	--------0----------------	---------0---------------	----------0--------------	-----------0-------------	------------0------------	-------------0-----------	--------------0----------	---------------0---------	----------------0--------	-----------------0-------	------------------0------	-------------------0-----	--------------------0----	---------------------0---	----------------------0--	-----------------------0-	------------------------0	1------------------------	-1-----------------------	--1----------------------	---1---------------------	----1--------------------	-----1-------------------	------1------------------	-------1-----------------	--------1----------------	---------1---------------	----------1--------------	-----------1-------------	------------1------------	-------------1-----------	--------------1----------	---------------1---------	----------------1--------	-----------------1-------	------------------1------	-------------------1-----	--------------------1----	---------------------1---	----------------------1--	-----------------------1-	------------------------1
Univariate occurrence	8	8	8	8	8	5	5	5	5	5	8	8	8	8	8	9	9	9	9	9	6	6	6	6	6	7	7	7	7	7	A	A	A	A	A	7	7	7	7	7	6	6	6	6	6	9	9	9	9	9
0------------------------	-	8	8	8	8	3	3	3	3	3	5	5	5	5	5	5	5	5	5	5	3	3	3	3	3	0	0	0	0	0	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5
-0-----------------------	8	-	8	8	8	3	3	3	3	3	5	5	5	5	5	5	5	5	5	5	3	3	3	3	3	0	0	0	0	0	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5
--0----------------------	8	8	-	8	8	3	3	3	3	3	5	5	5	5	5	5	5	5	5	5	3	3	3	3	3	0	0	0	0	0	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5
---0---------------------	8	8	8	-	8	3	3	3	3	3	5	5	5	5	5	5	5	5	5	5	3	3	3	3	3	0	0	0	0	0	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5
----0--------------------	8	8	8	8	-	3	3	3	3	3	5	5	5	5	5	5	5	5	5	5	3	3	3	3	3	0	0	0	0	0	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5
-----0-------------------	3	3	3	3	3	-	5	5	5	5	4	4	4	4	4	3	3	3	3	3	1	1	1	1	1	2	2	2	2	2	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	4	4	4	4	4
------0------------------	3	3	3	3	3	5	-	5	5	5	4	4	4	4	4	3	3	3	3	3	1	1	1	1	1	2	2	2	2	2	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	4	4	4	4	4
-------0-----------------	3	3	3	3	3	5	5	-	5	5	4	4	4	4	4	3	3	3	3	3	1	1	1	1	1	2	2	2	2	2	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	4	4	4	4	4
--------0----------------	3	3	3	3	3	5	5	5	-	5	4	4	4	4	4	3	3	3	3	3	1	1	1	1	1	2	2	2	2	2	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	4	4	4	4	4
---------0---------------	3	3	3	3	3	5	5	5	5	-	4	4	4	4	4	3	3	3	3	3	1	1	1	1	1	2	2	2	2	2	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	4	4	4	4	4
----------0--------------	5	5	5	5	5	4	4	4	4	4	-	8	8	8	8	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	0	0	0	0	0	3	3	3	3	3	5	5	5	5	5
-----------0-------------	5	5	5	5	5	4	4	4	4	4	8	-	8	8	8	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	0	0	0	0	0	3	3	3	3	3	5	5	5	5	5
------------0------------	5	5	5	5	5	4	4	4	4	4	8	8	-	8	8	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	0	0	0	0	0	3	3	3	3	3	5	5	5	5	5
-------------0-----------	5	5	5	5	5	4	4	4	4	4	8	8	8	-	8	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	0	0	0	0	0	3	3	3	3	3	5	5	5	5	5
--------------0----------	5	5	5	5	5	4	4	4	4	4	8	8	8	8	-	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	0	0	0	0	0	3	3	3	3	3	5	5	5	5	5
---------------0---------	5	5	5	5	5	3	3	3	3	3	5	5	5	5	5	-	9	9	9	9	3	3	3	3	3	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	0	0	0	0	0	6	6	6	6	6
----------------0--------	5	5	5	5	5	3	3	3	3	3	5	5	5	5	5	9	-	9	9	9	3	3	3	3	3	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	0	0	0	0	0	6	6	6	6	6
-----------------0-------	5	5	5	5	5	3	3	3	3	3	5	5	5	5	5	9	9	-	9	9	3	3	3	3	3	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	0	0	0	0	0	6	6	6	6	6
------------------0------	5	5	5	5	5	3	3	3	3	3	5	5	5	5	5	9	9	9	-	9	3	3	3	3	3	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	0	0	0	0	0	6	6	6	6	6
-------------------0-----	5	5	5	5	5	3	3	3	3	3	5	5	5	5	5	9	9	9	9	-	3	3	3	3	3	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	0	0	0	0	0	6	6	6	6	6
--------------------0----	3	3	3	3	3	1	1	1	1	1	3	3	3	3	3	3	3	3	3	3	-	6	6	6	6	3	3	3	3	3	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	0	0	0	0	0
---------------------0---	3	3	3	3	3	1	1	1	1	1	3	3	3	3	3	3	3	3	3	3	6	-	6	6	6	3	3	3	3	3	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	0	0	0	0	0
----------------------0--	3	3	3	3	3	1	1	1	1	1	3	3	3	3	3	3	3	3	3	3	6	6	-	6	6	3	3	3	3	3	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	0	0	0	0	0
-----------------------0-	3	3	3	3	3	1	1	1	1	1	3	3	3	3	3	3	3	3	3	3	6	6	6	-	6	3	3	3	3	3	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	0	0	0	0	0
------------------------0	3	3	3	3	3	1	1	1	1	1	3	3	3	3	3	3	3	3	3	3	6	6	6	6	-	3	3	3	3	3	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3	0	0	0	0	0
1------------------------	0	0	0	0	0	2	2	2	2	2	3	3	3	3	3	4	4	4	4	4	3	3	3	3	3	-	7	7	7	7	5	5	5	5	5	4	4	4	4	4	3	3	3	3	3	4	4	4	4	4
-1-----------------------	0	0	0	0	0	2	2	2	2	2	3	3	3	3	3	4	4	4	4	4	3	3	3	3	3	7	-	7	7	7	5	5	5	5	5	4	4	4	4	4	3	3	3	3	3	4	4	4	4	4
--1----------------------	0	0	0	0	0	2	2	2	2	2	3	3	3	3	3	4	4	4	4	4	3	3	3	3	3	7	7	-	7	7	5	5	5	5	5	4	4	4	4	4	3	3	3	3	3	4	4	4	4	4
---1---------------------	0	0	0	0	0	2	2	2	2	2	3	3	3	3	3	4	4	4	4	4	3	3	3	3	3	7	7	7	-	7	5	5	5	5	5	4	4	4	4	4	3	3	3	3	3	4	4	4	4	4
----1--------------------	0	0	0	0	0	2	2	2	2	2	3	3	3	3	3	4	4	4	4	4	3	3	3	3	3	7	7	7	7	-	5	5	5	5	5	4	4	4	4	4	3	3	3	3	3	4	4	4	4	4
-----1-------------------	5	5	5	5	5	0	0	0	0	0	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	5	5	5	5	5	-	A	A	A	A	6	6	6	6	6	4	4	4	4	4	6	6	6	6	6
------1------------------	5	5	5	5	5	0	0	0	0	0	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	5	5	5	5	5	A	-	A	A	A	6	6	6	6	6	4	4	4	4	4	6	6	6	6	6
-------1-----------------	5	5	5	5	5	0	0	0	0	0	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	5	5	5	5	5	A	A	-	A	A	6	6	6	6	6	4	4	4	4	4	6	6	6	6	6
--------1----------------	5	5	5	5	5	0	0	0	0	0	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	5	5	5	5	5	A	A	A	-	A	6	6	6	6	6	4	4	4	4	4	6	6	6	6	6
---------1---------------	5	5	5	5	5	0	0	0	0	0	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	5	5	5	5	5	A	A	A	A	-	6	6	6	6	6	4	4	4	4	4	6	6	6	6	6
----------1--------------	3	3	3	3	3	1	1	1	1	1	0	0	0	0	0	4	4	4	4	4	3	3	3	3	3	4	4	4	4	4	6	6	6	6	6	-	7	7	7	7	3	3	3	3	3	4	4	4	4	4
-----------1-------------	3	3	3	3	3	1	1	1	1	1	0	0	0	0	0	4	4	4	4	4	3	3	3	3	3	4	4	4	4	4	6	6	6	6	6	7	-	7	7	7	3	3	3	3	3	4	4	4	4	4
------------1------------	3	3	3	3	3	1	1	1	1	1	0	0	0	0	0	4	4	4	4	4	3	3	3	3	3	4	4	4	4	4	6	6	6	6	6	7	7	-	7	7	3	3	3	3	3	4	4	4	4	4
-------------1-----------	3	3	3	3	3	1	1	1	1	1	0	0	0	0	0	4	4	4	4	4	3	3	3	3	3	4	4	4	4	4	6	6	6	6	6	7	7	7	-	7	3	3	3	3	3	4	4	4	4	4
--------------1----------	3	3	3	3	3	1	1	1	1	1	0	0	0	0	0	4	4	4	4	4	3	3	3	3	3	4	4	4	4	4	6	6	6	6	6	7	7	7	7	-	3	3	3	3	3	4	4	4	4	4
---------------1---------	3	3	3	3	3	2	2	2	2	2	3	3	3	3	3	0	0	0	0	0	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	3	3	3	3	3	-	6	6	6	6	3	3	3	3	3
----------------1--------	3	3	3	3	3	2	2	2	2	2	3	3	3	3	3	0	0	0	0	0	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	3	3	3	3	3	6	-	6	6	6	3	3	3	3	3
-----------------1-------	3	3	3	3	3	2	2	2	2	2	3	3	3	3	3	0	0	0	0	0	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	3	3	3	3	3	6	6	-	6	6	3	3	3	3	3
------------------1------	3	3	3	3	3	2	2	2	2	2	3	3	3	3	3	0	0	0	0	0	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	3	3	3	3	3	6	6	6	-	6	3	3	3	3	3
-------------------1-----	3	3	3	3	3	2	2	2	2	2	3	3	3	3	3	0	0	0	0	0	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	3	3	3	3	3	6	6	6	6	-	3	3	3	3	3
--------------------1----	5	5	5	5	5	3	3	3	3	3	5	5	5	5	5	6	6	6	6	6	0	0	0	0	0	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	3	3	3	3	3	-	9	9	9	9
---------------------1---	5	5	5	5	5	3	3	3	3	3	5	5	5	5	5	6	6	6	6	6	0	0	0	0	0	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	3	3	3	3	3	9	-	9	9	9
----------------------1--	5	5	5	5	5	3	3	3	3	3	5	5	5	5	5	6	6	6	6	6	0	0	0	0	0	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	3	3	3	3	3	9	9	-	9	9
-----------------------1-	5	5	5	5	5	3	3	3	3	3	5	5	5	5	5	6	6	6	6	6	0	0	0	0	0	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	3	3	3	3	3	9	9	9	-	9
------------------------1	5	5	5	5	5	3	3	3	3	3	5	5	5	5	5	6	6	6	6	6	0	0	0	0	0	4	4	4	4	4	6	6	6	6	6	4	4	4	4	4	3	3	3	3	3	9	9	9	9	-

Figure 5.7: Co-occurrence measurements using the set of final configurations in Figure 5.5. Note that for reasons of space, any counts equal to 10 are represented with an ‘A’.

	0------------------------	-0-----------------------	--0----------------------	---0---------------------	----0--------------------	-----0-------------------	------0------------------	-------0-----------------	--------0----------------	---------0---------------	----------0--------------	-----------0-------------	------------0------------	-------------0-----------	--------------0----------	---------------0---------	----------------0--------	-----------------0-------	------------------0------	-------------------0-----	--------------------0----	---------------------0---	----------------------0--	-----------------------0-	------------------------0	1------------------------	-1-----------------------	--1----------------------	---1---------------------	----1--------------------	-----1-------------------	------1------------------	-------1-----------------	--------1----------------	---------1---------------	----------1--------------	-----------1-------------	------------1------------	-------------1-----------	--------------1----------	---------------1---------	----------------1--------	-----------------1-------	------------------1------	-------------------1-----	--------------------1----	---------------------1---	----------------------1--	-----------------------1-	------------------------1
0------------------------	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-0-----------------------	1	-	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
--0----------------------	1	1	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
---0---------------------	1	1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
----0--------------------	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-----0-------------------	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
------0------------------	-	-	-	-	-	1	-	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-------0-----------------	-	-	-	-	-	1	1	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
--------0----------------	-	-	-	-	-	1	1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
---------0---------------	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
----------0--------------	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-----------0-------------	-	-	-	-	-	-	-	-	-	-	1	-	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
------------0------------	-	-	-	-	-	-	-	-	-	-	1	1	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-------------0-----------	-	-	-	-	-	-	-	-	-	-	1	1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
--------------0----------	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
---------------0---------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
----------------0--------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-----------------0-------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
------------------0------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-------------------0-----	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
--------------------0----	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
---------------------0---	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
----------------------0--	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-----------------------0-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
------------------------0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1------------------------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-1-----------------------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
--1----------------------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
---1---------------------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
----1--------------------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-----1-------------------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
------1------------------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-------1-----------------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
--------1----------------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
---------1---------------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
----------1--------------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-
-----------1-------------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	1	1	-	-	-	-	-	-	-	-	-	-
------------1------------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	1	1	-	-	-	-	-	-	-	-	-	-
-------------1-----------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	-	1	-	-	-	-	-	-	-	-	-	-
--------------1----------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-
---------------1---------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-
----------------1--------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	1	1	-	-	-	-	-
-----------------1-------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	1	1	-	-	-	-	-
------------------1------	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	-	1	-	-	-	-	-
-------------------1-----	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-	-	-	-	-	-
--------------------1----	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1
---------------------1---	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	1	1
----------------------1--	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	1	1
-----------------------1-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	-	1
------------------------1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	1	-

Figure 5.8: The associations that are formed, based on the strongest symbioses seen in Figure 5.7. Only non-zero associations are shown with a number, and for the algorithm implementation in this chapter, all associations are formed with strength 1 or 0 (‘all-or-nothing’). We can straightforwardly recognise the units that will be associated, since their univariate occurrence count is equal to their co-occurrence with one another. That is, if, in all final configurations that i is present, j is also present, their co-occurrence and univariate occurrence counts will be identical. This rule does not demand that any particular pair of units is discovered in *all* demes (if this were the case, we would see the univariate count of 15).

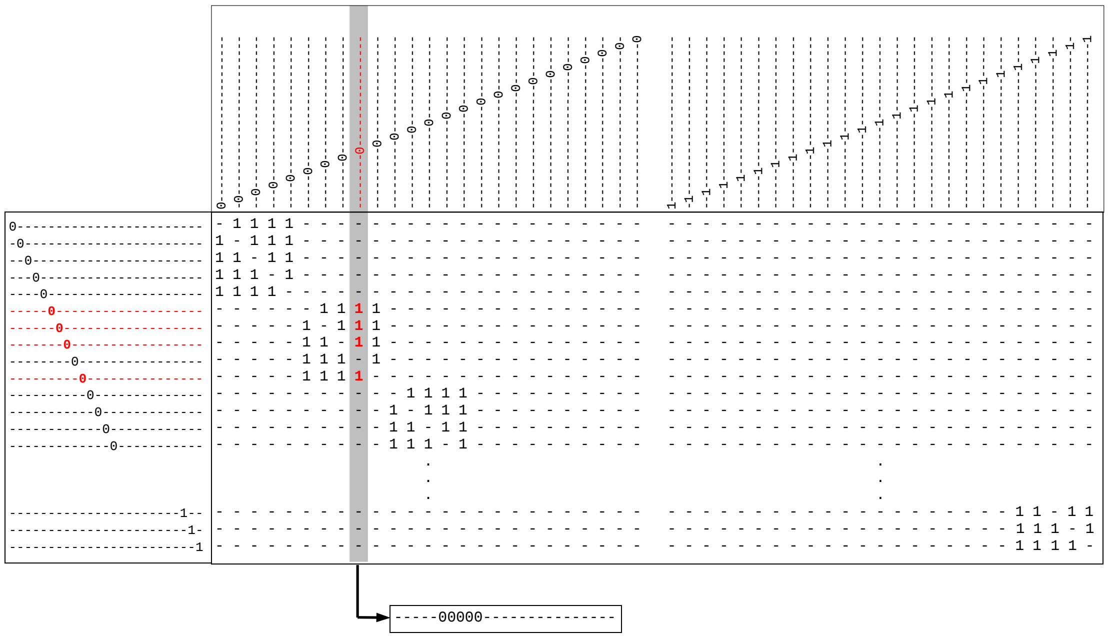
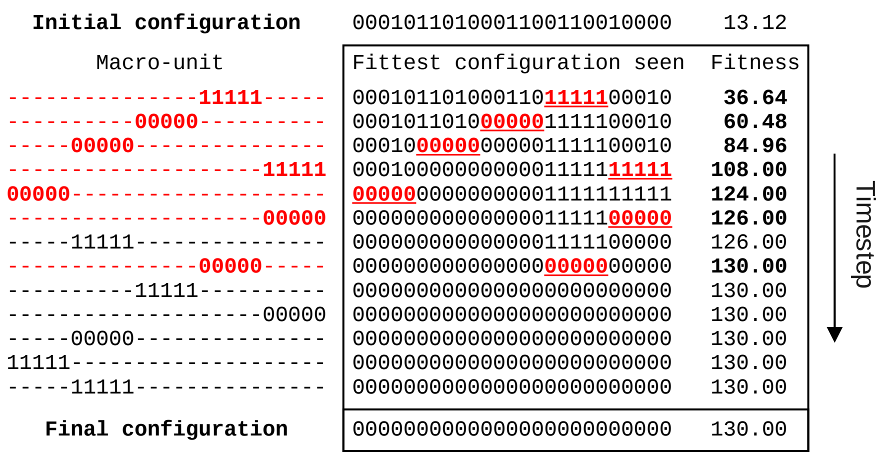


Figure 5.9: How the association matrix (Figure 5.8) is used to form higher-scale groups. For a given seed unit (highlighted in the column headings), the entries in the corresponding column describe the probability that other units will be included in the group (highlighted in the row headings). Since the algorithm in this chapter only assigns strengths of 0 or 1, group formation is deterministic.

```
00000--------------------
-----00000---------------
----------00000----------
---------------00000-----
--------------------00000
11111--------------------
-----11111---------------
----------11111----------
---------------11111-----
--------------------11111
```

Figure 5.10: The group formation process leads to a reduced set of specific variable settings. Since the associations are ‘all-or-nothing’ (Figure 5.9), the same groups are formed regardless of which member is taken as the group seed. In this example, we can reduce the set from 50 atomic-scale units (that specified the value for one variable) to 10 macro-scale units (each specifying values for 5 variables) – as shown in this figure.



Initial configuration	0001011010001100110010000	13.12
Macro-unit	Fittest configuration seen	Fitness
---------------**11111**-----	000101101000110**11111**00010	**36.64**
----------**00000**----------	0001011010**00000**1111100010	**60.48**
-----**00000**---------------	00010**00000**000001111100010	**84.96**
--------------------**11111**	00010000000000011111**11111**	**108.00**
00000--------------------	**00000**00000000001111111111	**124.00**
--------------------**00000**	00000000000000011111**00000**	**126.00**
-----11111---------------	0000000000000001111100000	126.00
---------------**00000**-----	000000000000000**00000**00000	**130.00**
----------11111----------	0000000000000000000000000	130.00
--------------------00000	0000000000000000000000000	130.00
-----00000---------------	0000000000000000000000000	130.00
11111--------------------	0000000000000000000000000	130.00
-----11111---------------	0000000000000000000000000	130.00
Final configuration	0000000000000000000000000	130.00

Figure 5.11: Hill climbing using the macro-scale units constructed from Figure 5.10. The availability of multi-locus units makes the search far more efficient, and in only a few steps a global optimum is discovered. After this point, none of the macro-units from Figure 5.10 can make a difference to the configuration. Note that the fourth macro-unit confers a significant fitness improvement when introduced, but it is ultimately replaced. In the VSM, solving a module to either solution is a large improvement over a random configuration in that module. Once all modules are at a module optimum, the between-module interdependencies provide a selection pressure to find agreeing module solutions. In this example, when the sixth macro-unit is introduced, it increases the number of modules that agree. Note that the change is only possible if the exact 5 variables are all changed at once: search with atomic-scale units would not be able to improve the module solution. MACRO is capable of solving the VSM problem reliably in the second epoch. However, more generally a set of final configurations from hill climbing with macro-units can be fed into further co-occurrence analysis (Figure 5.7) and association formation (Figure 5.8) for higher-scale search.

5.2.1 Representation

Our algorithm shares the objective of black-box optimisation with more classical evolutionary algorithms, although the representation we use deviates somewhat from other algorithms. In this chapter, as throughout this thesis, we consider fixed length problems with N binary variables.

Niches simply correspond to problem variables, and at the outset a species represents no more than a possible setting (allele) of that variable (locus). Thus, in our ecosystem there are $2N$ species, each occupying a particular niche. Each species can also have associations with other species. These associations are stored in a $2N \times 2N$ matrix S, where a row corresponds to all the associations that a species has with other species. That is, $S_{i,j}$ gives the strength of association from species i to species j. These values are initialised to zero, and remain in the range $[0, 1]$, and can be interpreted as probabilities of interaction. Self associations, $S_{i,i}$, are not used algorithmically.

This representation allows composite units to be formed from associated groups of atomic units. These composites are in general partial specifications of the entire problem, and need not be permanent groupings. However, in the specific algorithm in this chapter, when associations are made, they result in permanent higher-scale units.

Representing each allele with a species, rather than a species for each locus allows for the possibility of asymmetric associations between different alleles at the same locus and a third allele at another locus. Additionally this representation allows the application to problem domains where different assignments or solution components do not obviously compete directly for a particular position (*i.e.*, where niches are not defined).

5.2.2 Within-Deme Dynamics

In each of the demes, a different, random initial condition is constructed. In each timestep, the deme configuration can change by means of new species 'migrating' into the deme, potentially replacing the current occupants of any relevant niches. In general, these events can introduce multiple new species (*a migrant group*), according to their symbiotic associations. The process of forming such groups is illustrated in Figure 5.9. When no associations exist, migrant groups are simply single species, and consequently following the within-deme dynamics is very straightforward. In each timestep, a random species from the full ecosystem migrates into the deme, and temporarily displaces the current occupant of its corresponding niche. If the overall fitness of the deme is increased, the migrant remains. Otherwise, the deme reverts to its previous configuration. This procedure is simply following some form of adaptive walk specified by the fitness function, and is described in Table 5.1. Note that without any nonzero associations, groups consist only of the randomly selected seed species m (step 3).

Table 5.1: Association Informed Within-Deme Dynamics

```
Within each deme,
1.  initialise to a random combination of species that has every niche occupied;
2.  evaluate this initial combination; ($\rightarrow f_p$)
3.  update the current ecosystem configuration:
        (a) select, without replacement, a random migrant species, $m$
        (b) construct migration group $g$ with seed species $m$, according to Table 5.2
        (c) temporarily allow group $g$ to take precedence over all niches that it
    specifies;
4.  evaluate the modified species combination ($\rightarrow f_m$)
5.  if $f_m \geq f_p$
        allow all members of migrant group $g$ to remain permanently, and set $f_p \leftarrow$
    $f_m$
6.  if evaluation count not reached, go to step 3).
```

Following the deme dynamics will lead to combinations of species (*i.e.*, configurations of problem variables) that are locally optimal. The particular local optimum discovered will depend on the initial combination, and potentially the order in which migrant species are introduced. Note that in this algorithm exactly one migrant is introduced before evaluating the new ecosystem combination. While this is not central to the algorithm, introducing exactly one migrant between evaluations, rather than using a low probability of migration does keep multi-species migrations more straightforward in this and later chapters. In Section 5.3 we examine the difference between sampling with and without replacement.

5.2.3 Association-Informed Within-Deme Dynamics

Given a non-empty symbiosis matrix, we modify the deme dynamics such that those symbiotic relationships inform the behaviour. Associations that exist between species have the potential to lead to groups of species migrating at once. At the outset, species could only migrate individually before an evaluation is performed to determine if that migrant would remain. Nonzero symbiosis values lead to multi-locus changes in the deme dynamics. The strength of associations biases the distribution of occurrence of the possible multi-species groups. Specifically, one focal migrant species, m, is selected, and the corresponding row in the symbiosis matrix S defines which other species will make up the rest of the migrant group g. The S values are interpreted as probabilities of co-migration. The algorithm for constructing a migration group is as described in Table 5.2. Note that the association strengths can only be zero or one within the algorithm in this chapter. Accordingly, a migrant group constructed from a particular focal species will be the same every time. In later chapters, we generalise the situation to allow associations of intermediate strength.

Each problem variable (niche) can take multiple different values (species). Thus, a

Table 5.2: Procedure for constructing multi-species migration groups

```
Given a migrant species $m$, and the symbiosis matrix $S$,
1.  Add species $m$ to the empty group $g$
2.  For all other species, $x$, in a random order
        (a) Generate a uniformly random number, $\phi \in [0, 1]$
        (b) If $\phi < S_{m,x}$
                Add $x$ to group $g$, unless that niche is already filled in $g$
```

potential conflict arises if a nonzero association exists from m to more than one of the species that compete for the same niche. We resolve this by giving priority to the first species that was sampled when forming groups – and since the order in which species are sampled is random, it should not bias the outcome.

The within-deme dynamics can exhibit multi-locus changes, whose correlations are defined by the associations that have evolved. Migrant groups (Table 5.1, step 3(b)) will be created according to the association matrix, and are selected on as a unit. That is, if the ecosystem utility is improved with the multi-locus change, the entire group is retained; and if not, the entire group is discarded.[1] The initial condition in each deme can also be constructed using the symbiosis matrix, by adding migration groups built from random seed species until every niche is occupied.

Although algorithmically the change from search with atomic units to search with higher-level units is straightforward, the modification to the variation neighbourhood can have a significant effect. The configurations that were locally optimal with the original neighbourhood may have direct paths to configurations of higher fitness with the updated neighbourhood. We illustrate this in Section 5.3. But now, we describe how to adaptively form symbiotic associations such that this new neighbourhood has an impact.

5.2.4 Updating the Symbiosis Matrix

In this chapter, we use a simple method to update the symbiosis matrix to allow an initial exploration of the abilities of this approach. In the subsequent chapter, we use a more elaborate update rule that allows associations of intermediate strength.

We aim to form associations between species that have significant epistatic dependencies, such that in subsequent evolution those species will co-vary. Therefore we only aim to form associations where species are compatible as part of a sub-solution: there should be *positive* dependencies between such species. The associations are key to compositional relationships used in our bottom-up approach.

[1]Note that there is no propagation of these values in the current implementation. A focal migrant species m is chosen, and it is the row $S_{m,j}$ that defines these probabilities – the associations of species added to the consortia do not bias the probabilities of introducing further species. We have no reason to complicate matters.

We have an ensemble of demes that will each run from different initial conditions to different local optima when faced with a rugged landscape. We could in principle base decisions on the formation of associations upon the path taken during the transient period as well as the final state of each deme, and additionally take into account the fitness values for all configurations tested. However, the final state is the most enriched source of information: no additional dependencies can be satisfied with single variable changes from these optima. Furthermore, the configurations that are visited on the transient to a particular optimum are likely to be highly correlated with the optimum itself (albeit noisy versions of the optimum).

The fitness information is valuable for finding local optima, but there are two main reasons that we do not use this information in the symbiosis update stage. First, assigning credit to partial configurations suitably is a nontrivial issue (Potter and De Jong, 2000). Without good indication of how much a given component contributed to a particular fitness value, it becomes meaningless to use the fitness of the entire configuration. Second, by biasing the strength of an optimum's influence on the change in symbiosis value, there may be issues with a loss of diversity. For example, if there are strong nonlinear bonuses from some configurations, their fitness may become too dominant in the update, without necessarily resolving the conflicts necessary to find global optima. Therefore we elect to use only the correlations in the final states of each deme, and during the association formation ignore all fitness information.

If we observe two species, i and j, both to be present in the optimum found by a particular deme, we might suspect those two species to have a beneficial relationship, or perhaps to have no fitness interactions. However, with just one locally optimal configuration, it might be the case that those two species actually have a negative fitness interaction that is frustrated in the particular context. But if across several demes we consistently observe i and j appearing, it becomes less likely that these species are co-occurring by chance, and less likely still that there is a negative fitness interaction. If for *all* demes where i is present, j is *also* present, and vice versa, we can then infer a significant interdependency. In this case, the evidence for a valuable symbiotic association is as strong as it can be, and thus it is sensible to form a join[2]. Note that this does not demand that i and j are both present in all demes, only that when either is present the other must also be present.

[2]With one caveat: that a sufficiently sized sample is taken. More on this in Section 5.4.

Table 5.3: Association update according to co-occurrence

```
For each pair of species, $i, j$,
1.  If the set $I$ is identical to $J$
        make an association between $i$ and $j$ by setting $S_{i,j} = 1$
2.  If any element of $I$ is not in $J$, or vice versa
        set $S_{i,j} = 0$
```

Table 5.4: Symbiotic Optimisation: Main Procedure

```
1.  Allow $d$ demes to run to their local optimum (see Table 5.1)
2.  Measure the co-occurrence between each pair of species within all deme optima,
    and reinforce/update symbiotic associations (see Table 5.3)
3.  Randomise each deme configuration and go to step 1.
```

This simple rule is used to form 'all-or-nothing' associations between species. Once developed, such an association causes the species involved to co-vary all the time – akin to a symbiogenic union. Therefore it is important to err on the side of caution, and demand that the evidence for an association is as strong as possible. The rules for changing symbiotic associations are described in Table 5.3, which depend on the following definitions:

> Let the set of final deme states at the end of an epoch be D^*, where $|D^*| = d$.
>
> Let the set of final deme states for which i is present be I, and the set of final deme states for which j is present be J, where $I \subseteq D^*$, $J \subseteq D^*$.

Putting these different components together, the overall algorithm is described in Table 5.4.

5.3 An Examination of Operation

In this section we explore how the symbiotic algorithm behaves on a stylised problem with modular construction, describing how it automatically identifies the structure in the search space, and subsequently exploits it by providing specific multi-locus variation.

5.3.1 A Scalable Building Blocks Problem

Watson and Jansen (2007) introduce a synthetic problem class where instances are constructed from several large modules, or 'building blocks'. Each module has two optima, one fitter than the other, both with equal sized basins (*i.e.*, the same number of initial conditions lead to each optimum). These are concatenated without inter-module dependencies to construct the full problem. The fitness contribution of a single module is defined in Equation 5.1.

$$f(x) = \begin{cases} k & \text{if } U(x) = k, \\ \frac{U(x)}{2} & \text{if } k > U(x) > \frac{k}{2}, \\ \frac{(k-U(x))}{2} & \text{otherwise.} \end{cases} \tag{5.1}$$

Given that each of Z modules has k variables, x is a configuration of variables within that module, and $U(x)$ is the unitation (number of variables set to '1'). Without loss of generality, the all-1 configuration is chosen to be the fitter solution in each module, which results in the global solution being the concatenation of these fitter module solutions to also be all-1. We refer to this problem as the scalable building blocks (SBB) problem in this chapter.

Since each module has 2 solutions, there are 2^Z locally optimal configurations in the entire space, only one of which is globally optimal: when all modules are set to all-1, and hence all loci are set to 1. Each of the local optima has an equally sized basin of attraction. The factors that compete to make this problem difficult are the size of each module (k) and the number of modules (Z). Increasing k makes the exact change needed to move from one local optimum to another less likely – a specific k loci must change at once. Increasing Z dictates an exponential increase in the number of local optima, making the single global optimum rarer. In this chapter we balance these factors by setting $Z = k = \sqrt{N}$.

There are several reasons why we choose this problem to illustrate our new algorithm. First, the straightforward construction makes interpretation clear. Second, the clear modular structure in the SBB problem provides ideal conditions to validate the operation of a decompositional approach. Third, the absence of interdependency between modules means that only two steps are needed to solve the problem: construct the two locally optimal module solutions for each module, and subsequently select the better solution for each module to find the optimal configuration. Finally, this problem is provably difficult for local search, which allows us to demonstrate that the hill-climber that underlies our algorithm before associations have formed is not alone capable of solving the problem.

5.3.2 Why the SBB is Difficult: Possible Control Algorithms

In this section, we elaborate on the difficulties that the SBB problem presents. We work through a number of simple techniques in order to understand why they will fail to solve the problems efficiently.

A restart hill-climber that modifies a single bit at a time scales exponentially with the number of modules, as 2^Z. This is because there are 2^Z local optima, each of which must be considered to reliably find the fittest.

An alternative is to use a selective unit of entire ecosystems (*i.e.*, N-bit configurations). This does not allow one ecosystem configuration to have any correlation with the next, and hence cannot follow fitness gradients. Therefore, all possible configurations must be enumerated, which scales less favourably than enumerating just the local optima ($\mathcal{O}\left(2^N\right)$ versus $\mathcal{O}\left(2^Z\right)$).

In principle, migrations of uncorrelated groups of species can move between the local optima that defeat single-migrant dynamics. However, to move between local optima in the SBB landscape, the k species that are in one module must all change at once. There are a total of $2^k \cdot \binom{N}{k}$ different possible k-bit modules. Thus, any algorithm that attempts to use exactly the right sized migration groups, but without knowledge of specific group membership and hence forming k-species groups at random instead would have a vanishingly small probability of forming even a single module correctly (Watson and Jansen, 2007).

Any groups that do not replace all of the species within a module at once cannot escape from a local optimum to a configuration of higher fitness (although groups of size $1 < |g| < k$ can change the optimum discovered early on in the trajectory within one deme). Therefore migration groups must constitute entire module solutions in order to reliably find the global optimum.

Even if the decomposition were known, randomly forming a group of the particular membership required is still exponential in the module size (*i.e.*, $\mathcal{O}\left(2^k\right)$, and recall that in the SBB, $k \propto N$).

In summary, enumerating the 2^Z local optima is the fastest of these control algorithms (when k=Z), despite it not requiring any structural information. Therefore, any reasonable control that does not have some mechanism for decomposing the search space will not be capable of solving problems such as the SBB in less than time exponential in the system size.

5.3.3 How the SBB Could be Easy: Correlated Modular Variation

Let us for a moment set aside the issue of how to identify a suitable decomposition, and consider what a sensible target decomposition is. Suppose that we knew both a) which variables constitute a module, and b) which variable allocations were optimal. If we could vary entire module solutions as single components, the SBB problem would become trivial: effectively, a 'module-max', akin to a onemax problem that simply needs the correct module solution selected independently for each module.

MACRO-H uses a pairwise matrix to represent associations. The associations that reflect this decomposition would be strong between all of the 1s in each of the modules, as this would cause the entire module solution to vary as a component. As a minimal representation, this is all that is necessary. However, in problems that are not separable, but exhibit modular interdependency (such as the VSM, described in Chapter 3), both of the module optima would be advantageous, as the correct module solution depends on how the rest of the system is solved. Therefore, we aim to form associations that reflect each of the module solutions (*i.e.*, strong associations between all 1s in each module, and strong associations between all 0s in each module, and no others).

As noted in the previous section, random guesses at either the variables that constitute a module, or which variable allocations are optimal within a module do not provide a scalable method for solving the SBB. However, next we explore the information that we can access simply from fitness evaluations, and how we can use that information to systematically discover a decomposition that turns the SBB into a module-max.

5.3.4 Hill-Climbing on the SBB Problem

A typical run of a hill-climbing algorithm will end up at one of the 2^Z locally optimal configurations, being extremely unlikely to have found the single global optimum (*i.e.*, with probability 2^{-Z}). As there are no between-module dependencies, the final state of a module only depends on the initial condition of variables within that module. Within a single module, the solution that a hill-climber will find is determined by where the initial condition lies with respect to the basin boundary. The boundary is defined simply by majority.[3] In one arbitrary local optimum, the number of modules that are solved with the fitter solution is binomially distributed as $\binom{N}{Z}$.

From an ensemble of random initial conditions, we will find several different local optima. Unless the number of initial conditions are sampled is exponential in Z, the global optimum will not be found reliably (and because $Z = \sqrt{N}$, the required sample size for local search to find the global optimum scales as a function of N). However, that does not mean that nothing useful can be learned from the resulting configurations. Consider the set of six example local optima from a 25-bit SBB problem provided in Table 5.5.

Within each of the modules, both solutions have been found by at least one of the demes. It does not take very many demes for this to be the case, since the number of initial conditions that lead to each of the module solutions is equal.

5.3.5 The Formation of Adaptive Associations

Given the target associations identified in Section 5.3.3, can we use the information that appears in the local optima to identify such associations? From the example set of

[3]For odd-sized modules all configurations are in one basin or the other. For even-sized modules, there are $\binom{k}{k/2}$ initial conditions that could end up at either optimum with equal probability. This is dependent on the first migration that *changes* the deme configuration.

Example	Configuration	fitness
1	00000 11111 11111 11111 00000	20.0
2	00000 11111 00000 00000 00000	15.0
3	11111 00000 11111 00000 00000	17.5
4	11111 11111 00000 11111 11111	22.5
5	11111 00000 11111 00000 11111	20.0
6	11111 00000 00000 00000 00000	15.0

Table 5.5: several configurations for a 25-bit SBB problem (one per row). The loci with epistatic dependencies are collected together and space-separated by module, as a visual aid. This information is not defined by the problem – the physical to epistatic map is random. That is, the algorithms that we test are not provided with information regarding which bits belong to which modules. Notice that the appearance of a particular allele is consistent with the context that it appears in, even though both alleles appear in each locus in at least one deme in this set.

local optima in Table 5.5, we note that final deme configurations provide a clean signal. Visually, there is a clear pattern. There are only two configurations worth considering in each module: all-0 and all-1.[4]

Let us consider three classes of possible association, which are formed according to the simple joining rule defined in Section 5.2.

1. *Within module co-occurrence of compatible alleles.* In the first module, the all-0 solution is found twice and the all-1 solution is found four times. The occurrence of 1-alleles in the first and second loci is identical, and so these species will become associated. Corresponding associations will form between each pair of species in this module. Similarly, the occurrence of each of the 0-alleles in this module is identical, and accordingly associations will form between each pair of species here too.

2. *Within module co-occurrence of conflicting alleles.* A 0-allele and a 1-allele within the same module never co-occur in the same context, even though both occur in some contexts. No within-module associations of this type will be formed in this example.

3. *Between module co-occurrence.* Consider the all-0 module-solutions for the 3rd and 4th modules. In example 2 and example 6, both co-occur. However example 3,4 and 5 provide contradictory cases such that not all occurrences of zeros in the third module are concomitant with occurrences of zeros in the fourth module. This

[4]Three visual aids that assist in our illustration, which cannot be assumed by any algorithm under test. First, we nominate all-0 and all-1 as module solutions, rather than a random target configuration and its complement. Second, we order the loci according to their epistatic linkage, rather than leaving the epistatic to physical linkage map random. Third, we separate the loci to indicate module boundaries.

is the case for any pair of module solutions, including all-0 with all-0, all-0 with all-1, and all-1 with all-1 (subject to a sufficient sample size).

In this example, the above three conditions describe the associations that are formed. These associations lead to specific migrant groups that comprise entire module solutions. As an external observer we know that the all-0 solution is not part of the global optimum, it is nevertheless the second fittest configuration that a module can take, and is a sensible set of associations. Moreover, all of the 0-species involved are compatible with one another, and the most appropriate action for each is to form strong associations that lead to a (suboptimal) module solution. The all-1 solution within a module is the fittest configuration and all species are shown to be compatible.

But what about between-module associations? Spurious correlations could lead to such an association with an unrepresentative sample. However, since there are no fitness dependencies between modules in the problem, any such association would be erroneous. One might not think that this type of association would be damaging, especially because associations between two all-1 modules, or two all-0 modules are sensible. However, since both module solutions are equally likely, the chance of associating conflicting modules is as high as associating compatible modules. Forming an association between conflicting modules would be disastrous, since this would permanently rule out the possibility of finding the global optimum. Therefore a sample size sufficient to avoid such association is important to correct operation.

5.3.6 Employing the Symbiotic Associations

The formed associations create a new fitness landscape by creating non-trivial multi-locus variation. When these are used as macro-scale units in the rescaled hill-climbing process, they transform the problem into the module-max as described in Section 5.3.3. Searching in the space of module solutions is very efficient, and the problem will be solved reliably in each deme in the second epoch.

5.3.7 On the Choice of Hill-Climbing Process

The alternatives to the hill-climbing procedure in our algorithm require different locally optimal configurations to be discovered in different runs, cheap in terms of function evaluation, and straightforward rescaling it to allow multi-species variation. Many procedures fulfil these criteria, and here we consider two simple variants. When no associations have been formed, the process described in Table 5.1 is equivalent to hill-climbing without replacement. Using selection with replacement is an obvious alternative, so it is important to understand our rationale. The fitness over time of fifty repeats of each of these sampling protocols is plotted in Figure 5.12, on a 400-bit instance of the SBB

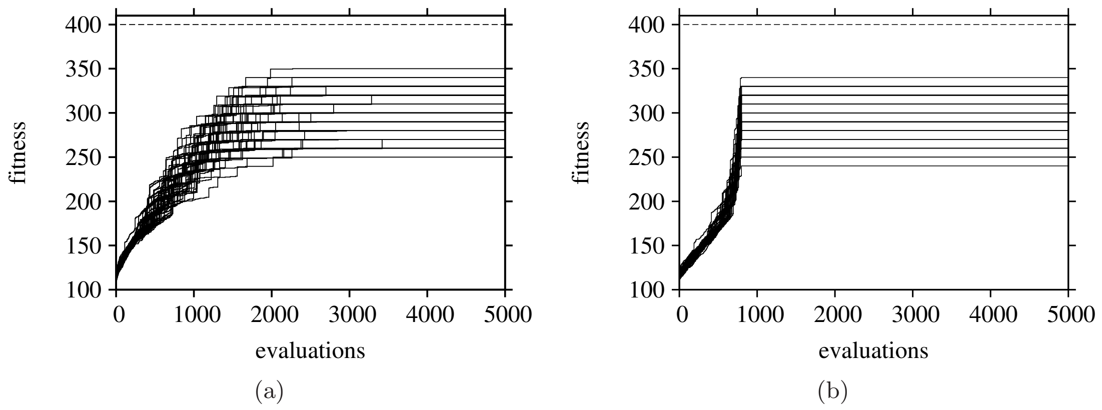


Figure 5.12: Fitness improvements from 50 repeats of two types of hill-climber on a 400-bit instance of the SBB problem. (a) shows a single-bit-flip hill-climber with replacement, and (b) shows a hill-climber without replacement. See text for explanation of the algorithms. Notice how for this stylised test problem, the hill-climber without replacement finds a local optimum within $2N$ evaluations, whereas the bit-flip hill-climber takes significantly longer. None of the runs find the global optimum (marked with a dashed line)

problem. All runs find a local optimum. The former algorithm finds a locally optimal configuration within $800 = 2N$ allele substitutions, whilst fitness improvements still occur up to approximately 3500 substitution attempts under the latter algorithm.

From a random initial configuration, we expect some allelic changes to be necessary to reach a local optimum. For a hill-climber that samples with replacement, the expected time to find a local optimum is dominated by correcting the last few alleles, and scales as $\mathcal{O}(N \log N)$ (Mühlenbein, 1992). However when sampling without replacement, within $2N$ samples all alleles will have been picked – scaling as $\mathcal{O}(N)$. In this problem, because there is no neutrality and no sign epistasis, no change in configuration that confers a fitness improvement will be reversed later in the same trajectory. Thus, it is sufficient to sample each allele once to reach a local optimum from any initial configuration.

5.4 Results

Here we investigate how the simple version of our symbiotic algorithm scales with problem size for two idealised building block problems: the SBB, as used in 5.3; and HIFF (the hierarchical if-and-only-if problem). We derive analytical expressions for the time complexity of this algorithm with respect to the problem size, and additionally provide results of simulated experiments.

5.4.1 Analytic Complexity on the SBB Problem

The form of our analysis is as follows. To demonstrate that the number of evaluations that MACRO-H requires to solve the SBB scales polynomially, there are three features that must be polynomial. If the number of evaluations used by each deme (τ); the number of demes used (d); and the number of epochs used (Υ) to solve the problem are all polynomial time, then the algorithm is polynomial for this problem.

MACRO-H is capable of solving the SBB problem in two epochs. In the first epoch, the initial search is for local optima in several demes. In the second epoch the search uses the associations developed to search in the space of module solutions, and a single macro-scale hill-climbing run is sufficient to find the global optimum.

As discussed in Section 5.3, we can be sure that a local optimum is reached after all alleles have been sampled (at least) once. We use sampling without replacement, which draws $2N$ samples, and therefore a trajectory costs $\tau = 2N + 1$ evaluations.

All that remains is to determine how many demes (*i.e.*, different initial conditions) must be used in the first epoch such that the correct solution to each module is found in at least one deme. Here we derive bounds for the number of repeats required to give a probability of success, s, greater than $1 - \epsilon$. We first derive a lower bound, and then take a branch from the analysis to derive an upper bound.

We define the probability that a single hill-climbing run finds the all-1 solution in one module as p, and the complementary event that the all-1 solution in that block is not found as $1 - p = r$. Only if every trial finds the all-0 solution will the algorithm fail overall. Therefore the probability of succeeding to find the all-1 solution in at least one trial using d demes, which we label as the event P, is:

$$\Pr(P) = 1 - \Pr\left(\text{miss } 1^k \text{ in } d \text{ trials}\right) = 1 - (1-p)^d = 1 - r^d. \tag{5.2}$$

Since the modules are independent, we can extend this to Z modules. We do not care which trial finds the all-1 solution for a given module, only that it is found *at least* once. Thus, the only case we need consider is failing that condition, which for each module is given by Equation 5.2. If for *any* module, the all-1 solution is not found, the algorithm will not be able to form all of the modules necessary to construct the global solution. Thus, the probability of success for Z modules, which we label s, depends geometrically on not missing any all-1 module solutions:

$$s = \left(1 - r^d\right)^Z. \tag{5.3}$$

By taking logarithms of both sides and rearranging,

$$\log(s) = \log\left(\left(1-r^d\right)^Z\right) = Z\log\left(1-r^d\right). \tag{5.4}$$

Therefore,

$$\log\left(1-r^d\right) = \frac{\log(s)}{Z}. \tag{5.5}$$

By the identity $\log(x) \leq x - 1$, we find $\log\left(1-r^d\right) \leq \left(1-r^d\right) - 1 = -r^d$. Therefore,

$$r^d \leq -\log\left(1-r^d\right) = -\frac{\log(s)}{Z}. \tag{5.6}$$

Taking logarithms of both sides,

$$d\log(r) \leq \log\left(\frac{-\log(s)}{Z}\right) = \log(-\log(s)) - \log(Z). \tag{5.7}$$

Noting that $Z = N^{\frac{1}{2}}$, letting $s = 1-\epsilon$, and dividing through by $\log(r)$ (which is negative since $r < 1$ and hence changes the sign of the inequality):

$$d \geq \frac{\log(-\log(1-\epsilon))}{\log(r)} - \frac{\log(N)}{2\log(r)}. \tag{5.8}$$

Equation 5.8 describes a lower bound on the number of demes required to solve the SBB problem, which scales asymptotically as $\log(N)$.

To derive an upper bound, we start from Equation 5.5 and manipulate in a different manner. The Taylor series expansion of $\log(1-x)$ is

$$\log(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \frac{x^4}{4} - \dots \tag{5.9}$$

Since $1 + x + x^2 + x^3 + \dots = \frac{1}{1-x}$, for $|x| < 1$,

$$\Rightarrow x^2 \cdot \left(1 + x + x^2 + x^3 + \dots\right) = \frac{x^2}{1-x}. \tag{5.10}$$

Let $r^d = x$. Hence, from Equation 5.9 and Equation 5.5,

$$\log(1-x) = -x - \left(\frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \dots\right), \tag{5.11}$$

$$> -x - \left(\frac{x^2}{1-x}\right), \tag{5.12}$$

$$> -x\left(1 + \frac{x}{1-x}\right). \tag{5.13}$$

The function $\frac{x}{1-x}$ is monotonically increasing in the interval $0 < x < 1$. Since $r < 1$ and

$d > 1$, $0 < r^d < 1$. From the same conditions, $r^d < r$. Thus, $\frac{x}{1-x} < \frac{r}{1-r}$. Hence,

$$\log(1-x) > -x\left(1 + \frac{r}{1-r}\right) = -x(1+C), \tag{5.14}$$

where $C = \frac{r}{1-r}$ is a positive constant.

Taking Equation 5.5, substituting in $s = 1 - \epsilon$, and resubstituting $x = r^d$,

$$\frac{\log(1-\epsilon)}{Z} = \log\left(1 - r^d\right) > -r^d(1+C), \tag{5.15}$$

$$r^d > -\frac{\log(1-\epsilon)}{(1+C)Z}, \tag{5.16}$$

$$d\log(r) > \log\left(-\frac{\log(1-\epsilon)}{(1+C)Z}\right). \tag{5.17}$$

Finally, we manipulate into a form that allows us a straightforward comparison with Equation 5.8, to describe d in terms of N (keeping in mind that $Z = N^{\frac{1}{2}}$). Note that dividing through by $\log(r)$ (which is negative since $r < 1$) changes the sign of the inequality:

$$d < \frac{\log(-\log(1-\epsilon))}{\log(r)} - \frac{\log(1+C)}{\log(r)} - \frac{\log(N)}{2\log(r)}. \tag{5.18}$$

Comparing Equations 5.18 and Equation 5.8, we see that the difference between lower and upper bounds is the problem-dependent constant term. The number of demes required is asymptotically bounded above and below by $\log(N)$.

Therefore the overall time required for MACRO-H to solve an SBB instance, in measured in function evaluations, is $\Upsilon \cdot \tau \cdot d$:

$$T_{\text{SBB,MACRO-H}} = 2 \cdot (2N+1) \cdot \Theta(\log(N)) = \Theta(N \log N). \tag{5.19}$$

For completeness, we use the specific value of $r = 0.5$ for the SBB problem, and hence $C = \frac{r}{1-r} = 1$. This yields the specific bounds on d:

$$d \geq \frac{\log(N)}{2\log(2)} - \frac{\log(-\log(1-\epsilon))}{\log(2)}, \tag{5.20}$$

$$d < \frac{\log(N)}{2\log(2)} - \frac{\log(-\log(1-\epsilon))}{\log(2)} + 1. \tag{5.21}$$

For comparison, we consider the expected number of evaluations required for an equivalent algorithm but without the ability to form associations. Each run costs $2N + 1$ evaluations. The order of initial conditions that must be sampled is equal to the number of local optima, since there is only one global optimum amongst them. There are $2^Z = 2^{\sqrt{N}}$ local optima, and so overall:

$$T_{\text{SBB,HC}} = (2N+1) \cdot \left(2^{\sqrt{N}}\right) = \Omega\left(N \cdot 2^{\sqrt{N}}\right). \tag{5.22}$$

The result in Equation 5.19 demonstrates that MACRO-H scales very efficiently with increasing problem size in the SBB, whereas hill-climbing scales exponentially with the problem size (Equation 5.22).

5.4.2 Simulated Experiments on the SBB Problem

Here we confirm the trend derived in Section 5.4.1 for scalability of the discrete algorithm on the SBB problem. We also compare it to its component mechanisms, to verify that forming associations provide a useful addition to the algorithm.

In this experiment, MACRO-H is applied to the SBB problem. The number of demes used is calculated from Equation 5.20, these values give at least 99% success rate over 100 repeats for each problem size tested. We compare this against the equivalent hill-climber that can only search at the micro-scale, on account of having no mechanism to inform macro-scale variation. This also uses $2N$ samples before restarting from a new random initial condition, and continues until the global optimum is discovered.

Figure 5.13 reports the mean number of evaluations required to find the global optimum for these algorithms. Standard error bars are displayed but are negligible with respect to the markers in most cases. In addition to the experimental data, we plot the analytical expressions given by Equation 5.19 and Equation 5.22, as derived in the previous section.

The expected trends for each of these algorithms fit the experimental data well, accepting some noise in the results from stochastic simulation. As discussed in Section 5.3.2, this problem is provably difficult for local search. An unshuffled version of this problem (*i.e.*, allowing algorithms to assume tight linkage) was shown to be solvable in $\mathcal{O}\left(N^2 \log N\right)$ evaluations by certain classes of genetic algorithm (Watson and Jansen, 2007). Here we use a shuffled SBB problem, and demonstrate that a log-linear number of evaluations is sufficient to solve this problem. Our result does not require any structural information to be provided *a priori*, instead discovering this information on the fly.

To put in perspective the value of the results in this section, we refer to the goals of this thesis (as laid out in Section 1.1). We stated our goals in two categories: to develop an algorithm that is capable of automatic problem decomposition for efficient optimisation; and to understand the mechanisms that give rise to any such efficiency. The development of MACRO-H satisfies the first goal: our approach is able to identify and subsequently exploit problem structure by searching in the space of combinations of lower-level sub-solutions that it has encapsulated. This process is very efficient, as we have demonstrated. The analytical derivation in Section 5.4.1 provides evidence that we have a deep understanding of how MACRO-H is able to identify unknown modular structures. Of course, we can further satisfy both of these goals, as we do in subsequent work in this thesis, but the contribution made here goes a long way to accomplishing our objectives.

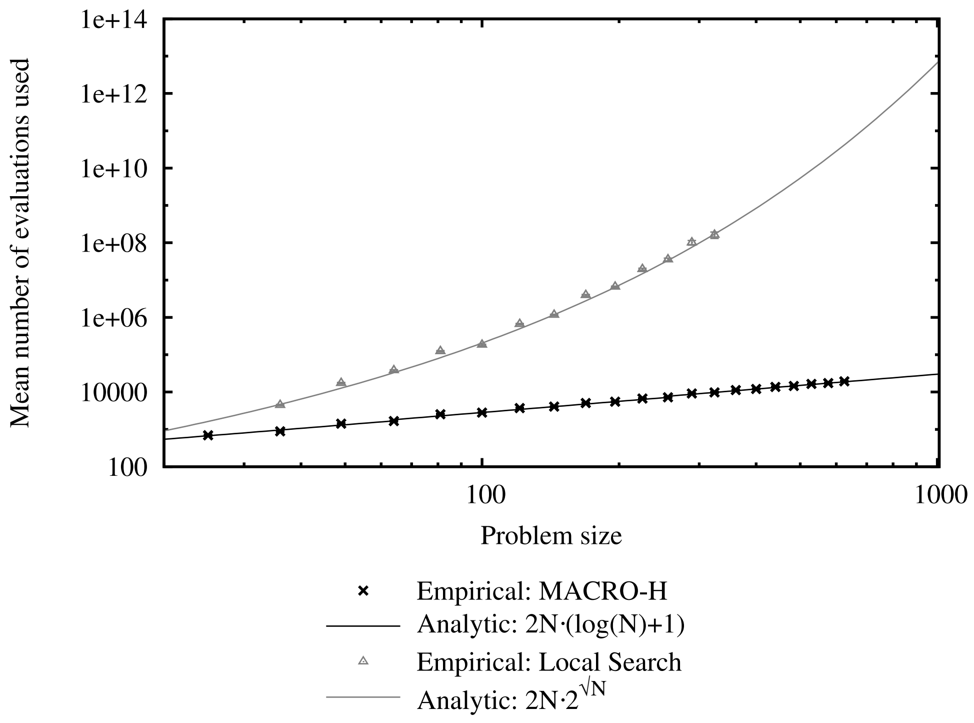


Figure 5.13: Results for MACRO-H on the SBB problem, with the global optimum found in at least 99% of 100 repeats. A random-restart version of the hill-climbing mechanism that is a component of each of these implementations is also tested. Note the sub-quadratic time complexity of both symbiotic optimisation methods on this log-log plot. Data points are measured, while the lines plotted use analytical expressions, as given in the legend (see also Equations 5.19 and Equation 5.22)

5.4.3 Analytic Complexity on HIFF

Hierarchically structured systems have the capacity to present significant difficulty to fixed variation mechanisms, and offer a validation check for the correct functioning of a recursive problem solving approach. The hierarchical if-and-only-if (HIFF) problem is a canonical instance of the class of hierarchically consistent problems, described in (Watson et al., 1998). Each layer of the problem has several modules. The recursive fitness contribution function is specified by Equation 5.23:

$$f(B) = \begin{cases} 1 & \text{if } |B| = 1, \\ |B| + f(B_L) + f(B_R) & \text{if } |B| > 1 \text{ AND } (\forall i : b_i = 0 \text{ OR } \forall i : b_i = 1), \\ f(B_L) + f(B_R) & \text{otherwise.} \end{cases} \tag{5.23}$$

Where B_L is the left hand component (from $x_0 \ldots x_{m/2-1}$) and B_R is the right hand component (from $x_{m/2} \ldots x_{m-1}$). This recursive definition allows for partial evaluation (*i.e.*, when $m < N$), but throughout this thesis we use it in a black-box manner (*i.e.*, the function only returns a fitness value for fully-specified candidate solutions).

HIFF has $2^{\frac{N}{2}}$ optima, two of which are globally optimal. For an N variable problem there are $L = \log_2 N$ layers in total. The number of modules in layer l is $N/2^l$, and thus in the entire problem there are m_{HIFF} modules:

$$m_{\text{HIFF}} = \sum_{l=1}^{L} \frac{N}{2^l} = N \sum_{l=1}^{L} \frac{1}{2^l} = N\left(1 - \frac{1}{2^L}\right) = N - 1. \tag{5.24}$$

As for the SBB problem, $2N$ steps are sufficient to find a locally optimal configuration. However, unlike the SBB, the overall function is dependent on all of its variables. Moreover, the optima at each hierarchical level are maximally distant in Hamming space, creating order-N dependencies at the highest level. This means that HIFF is pathologically difficult for a mutation-only hill-climbing process to solve (Watson et al., 1998).

The analysis here follows the analysis from Section 5.4.1, although recursion is used to handle the layers of the hierarchical structure. Let us assume that we are interested only in finding one global solution, and without loss of generality we will derive the probability for successfully finding the all-1 global solution. As for the SBB problem, we must have a sufficient number of demes to find the all-1 solution in each module at least once across all demes. Note that micro-scale hill-climbing can only follow fitness gradients to module solutions at the lowest level of the problem to start with. Modules consist of two bits, so for an N variable problem, $Z = \frac{N}{2}$. Once the first layer is solved and symbiotic associations have been formed, the number of modules at the second layer is halved to $N/4$. Hill-climbing in this space of reduced dimensionality will find each module solution with $p = 0.5$, and $r = 1 - p = 0.5$. This process continues as each layer is solved. If the number of demes selected is sufficient to reliably solve the first layer it will be sufficient for all higher layers. We manipulate Equation 5.7 and Equation 5.17 in the same manner as for the SBB analysis, with $Z = N/2$, to arrive at lower and upper bounds for the required number of demes, d, for the first layer in HIFF:

$$d \geq -\frac{\log(N)}{\log(r)} - \frac{\log(2)}{\log(r)} + \frac{\log(-\log(1-\epsilon))}{\log(r)}, \tag{5.25}$$

$$d < -\frac{\log(N)}{\log(r)} - \frac{\log(2)}{\log(r)} + \frac{\log(-\log(1-\epsilon))}{\log(r)} - \frac{\log(1+C)}{\log(r)}. \tag{5.26}$$

For our analysis of MACRO-H on SBB problems, we make the simplifying assumption that τ has fixed cost in each layer. Since it takes several epochs to solve a hierarchical problem such as HIFF (as opposed to two epochs for an arbitrarily sized SBB instance), this assumption becomes relevant. Specifically, the assumption made above is that at each of $L = \log_2 N$ layers, d hill-climbing passes must be made, each at cost $2N + 1$ evaluations. However, since the hill climbing need only consider introducing each of the (macro-)units that exist in the system once per pass, and the number of units reduces as a function of the current layer. Let us define E as the number of units available in the ecosystem, such that $E = 2N$ at initialisation, and decreases as the dimensionality is reduced. When solving a problem such as HIFF, E is halved at every epoch, as a layer of the problem is solved. Therefore, the cost in evaluations of MACRO-H on HIFF is given by:

$$T_{\text{HIFF,MACRO-H}} = \sum_{l=0}^{L} d \cdot \tau = \sum_{l=0}^{L} d \cdot \left(\frac{2N}{2^l} + 1 \right), \tag{5.27}$$

$$= 2dN \cdot \sum_{l=0}^{L} \frac{1}{2^l} + \sum_{l=0}^{L} d, \tag{5.28}$$

$$= 2dN \cdot \left(2 - \frac{1}{N} \right) + d\,(L+1)\,, \tag{5.29}$$

$$= 4dN + dL - d. \tag{5.30}$$

This expression is asymptotically dominated by the $4dN$ term, and since d is bounded asymptotically both above and below by $\log(N)$, we find that the overall requirement is given by:

$$T_{\text{HIFF,MACRO-H}} = \Theta(N \log N). \tag{5.31}$$

We also provide a comparison for the equivalent algorithm with the ability to form associations. From each initial condition it takes $2N + 1$ evaluations to reach a local optimum, and the number of basins that must be sampled is the the number of local optima over the number of global optima. This results in an expected waiting time of:

$$T_{\text{HIFF,HC}} = (2N+1) \cdot \left(2^{\frac{N}{2}-1} \right) = \Omega\left(N \cdot 2^{\frac{N}{2}} \right). \tag{5.32}$$

Finally, we note that although this analysis is derived with HIFF in mind, it can be applied to other hierarchical problems including the hierarchical trap as defined in (Pelikan and Goldberg, 2000). This uses 3-bit traps in a hierarchical manner, and hill-climbing will find the fittest (all-1) solution in each module with $p = 1/4$. Although the constants are different, the complexity class is the same as for HIFF.

5.4.4 Simulated Experiments on HIFF

Here we confirm the trend derived in Section 5.4.3 for scalability of the discrete algorithm on the HIFF, and compare it to its component mechanisms.

We apply the symbiotic optimisation algorithm to a variety of sizes of HIFF instances. The number of demes used for each problem size are given in Table 5.6, and these provide a success rate of at least 99% in 100 runs for each problem size tested. The

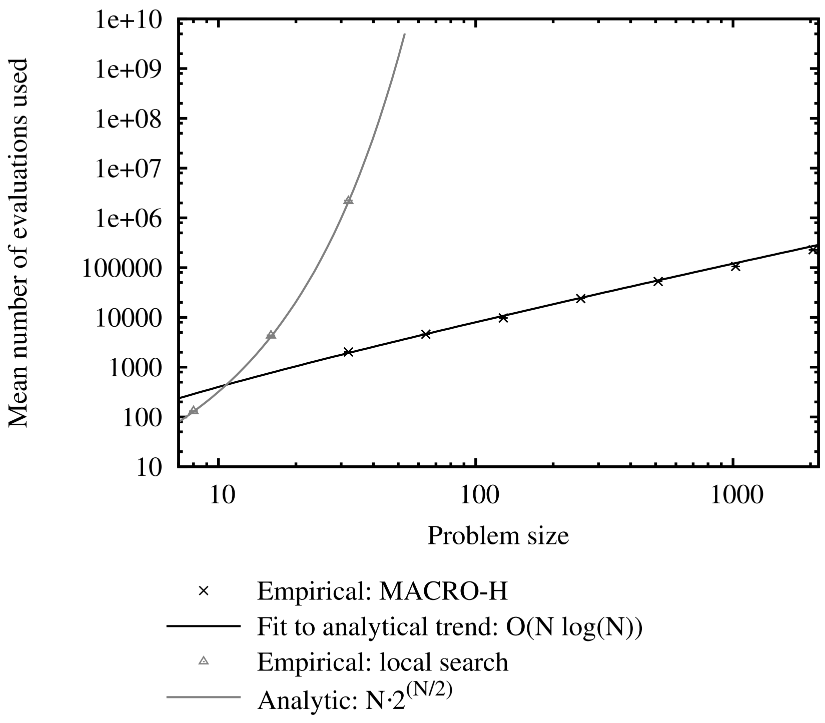


Figure 5.14: Results for MACRO-H on HIFF, with the global optimum found in at least 99% of 100 independent runs. This algorithm is able to solve large instances (up to 2048 variables tested here), whereas for local search methods this problem is clearly intractable. Data points are from empirical tests, a line of the form $\mathcal{O}(N \log N)$ is fitted to the MACRO-H result, according to Equation 5.31, and an analytical prediction is plotted for the local search according to Equation 5.32.

equivalent micro-scale hill-climbing algorithm with no associations is also compared here, which restarts from new random initial conditions until a global optimum is found. On account of the exponential time required by this algorithm (see Equation 5.32), it is computationally infeasible to collect experimental results for $N > 32$.

Figure 5.14 displays the mean number of evaluations required to find the global optimum for these algorithms. The mean evaluation counts are only given for the successful runs, and error bars are displayed but are negligible with respect to the markers in most cases. In addition to the experimental data, we plot the expected trend given by Equation 5.32, and a line with the order of the trend given by Equation 5.31 as derived above.

Problem size, N	32	64	128	256	512	1024	2048
analytic deme count (5.26)	17	18	19	20	21	22	23
demes used in Figure 5.14	20	20	20	24	26	26	28

Table 5.6: Demes used by MACRO-H to solve HIFF, providing a success rate of at least 99% of 100 runs. See Figure 5.14.

HIFF, like the SBB, also presents an exponential number of local optima and also causes severe difficulty for local search methods. Despite using a very simple associative framework, separating the timescales between exploiting the decomposition and adapting the decomposition enables MACRO-H to outperform state of the art Bayesian model building algorithms. Specifically, our analytic and empirical results show that MACRO-H scales as $\mathcal{O}(N \log N)$ on this hierarchically decomposable problem. This is superior to hBOA, which was shown to scale as approximately $\mathcal{O}\left(N^{1.62} \log N\right)$, matching closely to an analytical prediction of $\mathcal{O}\left(N^{1.55} \log N\right)$ (Pelikan, 2002). Note that the algorithm of Iclanzan and Dumitrescu (2007) is similar to MACRO-H and is also demonstrated to scale as $\mathcal{O}(N \log N)$ on this problem. As noted in Section 5.1, we show how the idealised approach of MACRO-H can be generalised to solve a broader class of problems in the next chapter.

5.5 Discussion

Using simple hill-climbing as part of our algorithm offers several benefits. The most straightforward improvements in fitness are realised, and the resulting final deme configurations exhibit more order than the initial conditions. While we cannot satisfy all of the dependencies between species in any one deme, hill-climbing leads to satisfying many of those dependencies. One alternative to parallel hill-climbing is to construct a set of random configurations and select a high-fitness subset. A comparable scheme would use a population size of $d \cdot 2N$, and select d individuals. Random configurations are most densely distributed about the lowest fitness configurations of each module, according to the Binomial distribution. These configurations are thus likely to manifest many conflicts, in comparison to the neatly defined final deme configurations discovered by hill-climbing, within which simple conflicts are frequently resolved. Since our algorithm aims to make associations between species that have fitness-dependent interactions, the initial micro-scale optimisation provides a good way to restrict the set of associations that are considered. When a pair of species co-occur in a local optimum, this indicates a viable compatibility. We turn a viable compatibility into a concrete association through the inspection of an ensemble of local optima.

The mean fitness of a selected subset from random configurations is very unlikely to be as high as the final deme configurations on account of the distribution of random configurations. Additionally, strong selection pressure for high mean fitness is liable to lead to convergence in patterns, without the use of some mitigating diversity maintenance scheme. Our approach of using several independent demes offers an explicit method to ensure diverse solutions from the search space. Starting from several random initial conditions means that many different basins of attraction are sampled. Within a particular basin, it is unnecessary to represent several configurations that lie close to the same local optimum: the optimum has the richest information. Ensuring that enough basins are sampled (by using enough demes) is key to successful operation, as we saw in Section 5.4.

Neighbourhood Change & Dimensional Reduction

Initially the state space has 2^N possible configurations, and 2^Z different local optima (for the hill-climbing process described in Table 5.1). When associations are formed such that species co-vary, the number of possible states that can be reached is reduced: there are fewer degrees of freedom. This also has the consequence of modifying which configurations neighbour one another. In particular, when groups corresponding to module solutions are specified, the neighbourhood of one of the configurations that was locally optimal in the original units contains other configurations that were previously locally optimal, and not configurations in between. Now, there are $2Z$ groups (each of size k), and hence the accessible state space is only 2^{2Z}. In this situation, competition is transferred to the level of the module. With macro-scale variation that corresponds to the module solutions, the hill-climbing dynamics of each deme operate in a neighbourhood that is drastically different from the original space. At the module level, the global optimum is trivially accessible from any initial condition (as described in Section 5.3). Note however that while searching in the space of modules is significantly reduced from the full system, it is still exponentially sized with respect to N. Therefore, a random search process is not sufficient to find the global optimum, even in this space. Instead, a systematic search must be used. Hill-climbing at the macro-scale is efficient, given the appropriate macro-scale units. Thus, we should see how the influence of the neighbourhood change is more important in problem solving than the dimensional reduction; it is the specificity of groupings that provides both neighbourhood change and reduced dimensionality.

5.5.1 Comparison with the RSSA

In developing the RSSA (introduced in Section 4.2), we had the same underlying motivation as in the present algorithm: to evolve symbiogenic relationships that reflect a system's decomposition, and exploit those relationships by means of correlated variation. Unsurprisingly, the RSSA and MACRO-H share some features, which include the representation; the change in ecosystem when symbiogenic joins evolve; and the recursive application.

1. The atomic units of single-allele species is equivalent, as are the permanent influences from symbioses.

2. Both methods recursively make explicit joins, and are thus appropriate for hierarchically structured problems, as well as those with flatter, but modular structures.

3. Because of the explicit dimensional reduction, relationships must be symmetrically applied (as is the case when maximising reciprocal synergy, and must be the case when considering correlations, since these contain no directional information). Note that this does not ensure that the benefit of a join is equal to both/all partners.

4. Both algorithms base their decisions of when to invoke a symbiogenic join on an aggregation of information from multiple contexts – to establish a baseline, and to ensure that a join decision is not biased too strongly by an unrepresentative sample. As implemented, this does prevent either algorithm from being localised/decentralised. However, as later chapters discuss, the modifications to do so are few and demonstrably feasible.

The methods have significant differences, in their clarity of operation, and consequently their efficiency (scalability). Major distinctions stem from the separation in timescales in MACRO-H that enable micro-scale search to guide adaptation in the decomposition. We highlight a number of distinctions below, considering MACRO-H with respect to the RSSA.

1. The current units are exploited before making any attempt to adapt the unit of variation.

2. Consequently, search at more than one scale is implemented through a separation of timescales of search in the current units and the modification to those units.

3. Specifically, (1) and (2) lead to locally optimal contexts (in contrast to arbitrary contexts used in the RSSA).

4. Local optima are significantly less noisy than random contexts (at least in the problems tested in this chapter, and should be true to the extent where system dynamics are contractive): the micro-scale hill-climbing process improves the signal-to-noise ratio of co-occurrence information. Patterns of co-occurrence at local optima indicate which species work well together; which sets of species are compatible.

5. Therefore, the decision on whether a pair/group of species should be associated is simple in MACRO-H. Specifically, a small number of contexts is sufficient, and the decision mechanism is straightforward and computationally inexpensive (inspecting a number of contexts for co-occurrence).

6. Contrast this with a difficult decision on noisy information in the RSSA, where many contexts are required, and the decision mechanism is much more expensive (a type of perturbation test is used to assess synergy, requiring further function evaluations between each pair of species).

7. A further consequence of using local optima is that the contexts are stable, *i.e.*, no single species/composite entity change will result in increased fitness. Therefore,

   - It is likely to require many random contexts to faithfully reveal if a species pairing is suited or not. This is in part because the possible reasons why an association looks beneficial are either an immediate fitness improvement from ecosystem configuration change, or an improvement from the associative change. Using a stable context (where no single perturbation is an improvement) is a more efficient way, because it rules out the first potential case.
   - By only considering relationship changes between species that could feasibly coexist in a stable environment (*i.e.*, a local optimum), the following issue is avoided. In the RSSA, a pair of species that compete for the same niche could nevertheless be considered for symbiogenic unification. This type of competitive relationship is not correctly characterised by 'symbiosis', but the synergy metrics frequently return a strong synergy value – avoiding joins of this type is problematic. In MACRO-H, this issue is avoided entirely.

5.5.2 Limitations

The extreme joining rule is appropriately conservative when making such extreme and irreversible joins between species, but it is perhaps apparent that if the success of the algorithm hinges on making joins, and that joining mechanism requires evidence that is entirely uncontradicted, then the mechanism may be brittle in some situations where perfect/noise-free information is not available. We investigate this in the next chapter.

In this chapter, we have shown improved performance using a similar variety of modular problems to the RSSA results (hierarchical, and single-scale modularity). Thus, it may appear that we have not explicitly demonstrated that MACRO-H has a widened applicability than the RSSA. However, although the forms of the SBB and the VSM are similar, in fact the VSM provides a nonlinear fitness bonus for every variable change that increases agreement within a module. The RSSA can therefore detect synergy which is sufficient to lead to joins. Conversely, in the predominant majority of contexts in the SBB, there is only a linear fitness improvement when increasing the number of variables that agree. For the RSSA to detect some synergy, all but two variables are set to 1, and the two species that make up the rest of the module solution must be being tested for a join. The likelihood of a context of this form is given by $\frac{\binom{Z}{Z-2}}{2^Z} = \frac{\frac{Z(Z-1)}{2}}{2^Z} = \frac{Z(Z-1)}{2^{Z-1}}$, which decreases rapidly as the module size increases. Therefore, demonstrating that MACRO-H can solve the SBB is in fact an advance in generality over the RSSA, in addition to the qualitative improvements in efficiency.

5.6 Summary

In this chapter we have introduced a simple algorithm that can evolve an appropriate level of modularity for the structure present in the search space, and consequently is able to exploit the discovered decomposition to search at the macro-scale over several levels of organisation.

The major conceptual advance is understanding where multiple scales of search can be effective, and how exploiting the current scale before identifying (evolving) a new scale of search can be more efficient than using either scale individually.

This conceptual advance is manifested in a very efficient algorithm on modularly structured problems, scaling as $\mathcal{O}(N \log N)$ on examples of both single-scale and hierarchical problems. Linkage information is not required for these results, since it is discovered by the algorithm through the development of symbiotic associations. This automatic problem decomposition provides correlated variation that can traverse rugged landscapes with ease.

The efficiency of this algorithm can be attributed to similar reasons as we discussed in Section 4.1: hill-climbing (by mutation) initially find module solutions, and different demes find different module solutions; associations (that will modify the subsequent local dynamics) correspond to module solutions – and so like crossover, the association-informed dynamics can compete at the module level. Both scales of search are required to solve such a modular problem, and these scales of search are achieved through separating the timescales upon which associations are reinforced from that where associations are exploited. However, a critical difference between the result obtained in this chapter and the results in Section 4.1 is the assumption of tight linkage. In this chapter, no linkage information is assumed: it is automatically discovered. In the previous chapter, crossover could only provide modular variation because the epistatic linkage information was provided *a priori*.

This chapter has demonstrated that this separation of timescales of ecosystem configuration changes from the associative changes can implement the desired matching of the scale of evolution to the scale of structure present in the landscape.

Chapter 6

Probabilistic Symbiotic Relationships

The algorithms developed in the previous two chapters used adaptive variation mechanisms inspired by symbiogenic encapsulation, where any change in relationship between species has the extreme position of a complete inseparable union. In this situation, the new search neighbourhood is well defined (and in biological terms, that new evolutionary units exist are uncontroversial; although the creation of these units is not addressed in these algorithms). Contrast this with the less extreme situation whereby symbiotic associations of intermediate strength can evolve. Here, the likelihood of species co-varying is increased (when compared to a freely-mixed situation), but those species still have a chance of existing without the other. This type of continuous association has the capacity to describe a richer class of epistatic relationship than a discrete association – but can it be useful in facilitating multi-scale search? This representation is richer because it does not discard all the information that is present when the contexts observed are not sufficient to make a conservative 'yes' decision to make a join. Just because the evidence is not strong enough to support a permanent and irreversible join, it does not mean that the information is not valuable and worth retaining.

In this chapter, we aim to determine if it is feasible to implement probabilistic association formation, and corresponding exploitation, within the MACRO framework. In particular, we address the following research question:

> The discrete method introduced in the previous chapter is elegant, minimalist, and makes direct use of the information in the correlations between species in the local optima present. A probabilistic method of representing symbioses of intermediate strength ought to be more broadly applicable when correlations are fuzzy. Can we demonstrate this?

To address this aim, we develop a further algorithm, which analyses the co-occurrence between species from an ensemble of local optima (in the current units) to inform the updated strength of symbiotic associations. Associations are updated according to the deviance of observed covariance from the expected covariance that is predicted from the observed univariate frequencies.

We find the new technique to be superior in cases where local optima are ambiguous (contexts in which selection on associations occurs do not exhibit as high a level of order as those that are local optima in the SBB for instance). We demonstrate that this change stems from contradictions that cast doubt on an extreme join – restricting the appropriateness for a conservative joining mechanism. We also show that the selective environment of a species can be significantly improved without the need for associations to lead to explicit encapsulation of other species. Significant correlations can evolve that form highly coherent multi-locus migration groups. This enables macro-scale search, without explicitly reducing the dimensionality of the search space.

6.1 On the Limitations of All-or-Nothing Joins

In Section 5.3, we saw how parallel hill climbers provide contexts that allow simple rules to successfully form adaptive associations. One contributing factor to this result is the highly ordered fitness landscape provided by the SBB problem, and consequently, the high level of regularity in the optima discovered by hill climbing at the micro-scale. However, if the local optima discovered were not so neat and the co-occurrence of particular species not so clear cut, such a simple rule set for association formation is unlikely to suffice. Specifically, rule (1) (see Table 5.3) demands that in order to make a join, species j is present for *all* occurrences of species i, and vice versa. Hence, just a single deme that had only one of i and j present without the other is sufficient to rule out the join.

Consider a situation where such mismatches occur frequently even between species that have a positive epistatic interaction. In this case, the efficiency in solving structured problems demonstrated in Section 5.4 would not be achievable. How then, can we construct a problem to investigate this situation further? And how can we provide more robust association formation decisions?

One general property that could provide this change is to incorporate uncertainty into the landscape. We elect to achieve this by introducing ambiguity as to which variable assignments contribute to local optima. We make this modification to the SBB problem.

6.1.1 Introducing Neutrality into the SBB Problem

We modify the basis function used in the SBB problem by introducing a small amount of neutrality at the module solution. For η bits of neutrality in a k bit module, the

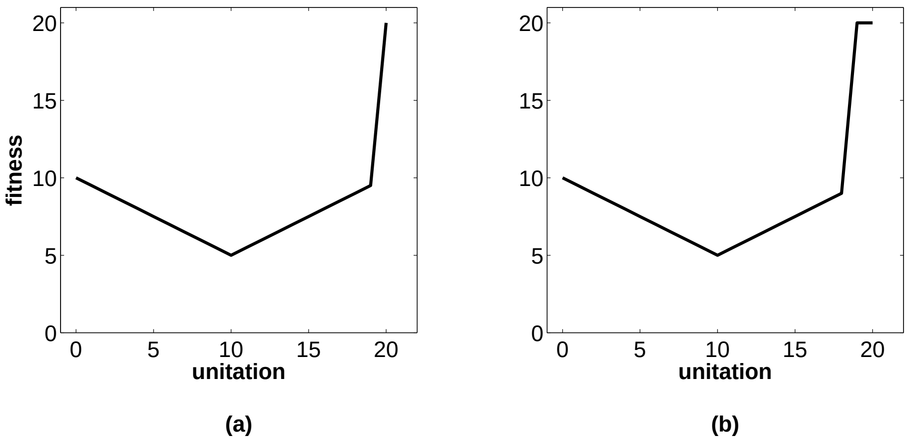


Figure 6.1: Fitness contribution according to unitation for one module of 20 bits, (a) for the SBB problem, and (b) for the SBB-N problem, with the level of neutrality, η, set to one bit.

fitness contribution is defined by Equation 6.1 (compare this to Equation 5.1).

$$f(x) = \begin{cases} k & \text{if } U(x) \geq k - \eta, \\ \frac{U(x)}{2} & \text{if } k - \eta > U(x) > \frac{k}{2}, \\ \frac{(k - U(x))}{2} & \text{otherwise.} \end{cases} \tag{6.1}$$

where x is a k-allele configuration, $U(x)$ is the unitation function, and $\eta < k/2$. Figure 6.1 compares the fitness contribution of a module from the SBB and SBB-N problems.

The overall fitness is given by a sum of the fitness contributions from each module, as for the regular SBB problem. We refer to this test problem as the scalable building block with neutrality, or SBB-N. As for other problems used in Chapters 5–7, the SBB-N has a shuffled linkage map.

Each SBB-N module has locally two optimal fitness values: $k/2$ for the all-0 solution, and k for any of the configurations that have at least $k - \eta$ ones. We refer to the latter class of configurations as the all-1 plateau. As for the SBB (see Section 5.3), it is still the case that half of the configurations will lead to each locally optimal fitness value under local search. However, for those in the all-1 plateau, the specific final configuration will not be identical in every case.

6.1.2 Contradictions in the SBB-N

For a module of k bits with $\eta = 1$ bit neutrality, there are $\binom{k}{\eta} + 1 = k + 1$ different genotypes in the all-1 plateau: each of the different permutations of $k - 1$ 1s and a single 0, in addition to the all-1 configuration.

For MACRO-H to form joins between all pairs within a module, all of the hill climbers in that epoch must find the same configuration. This becomes increasingly unlikely with larger modules or deme count. The example set of optima (for one module) shown in Table 6.1 highlights the problem.

Example	configuration						associations possible	
	1	2	3	4	5	6	1-species indices	count
1	1	1	1	1	1	0	–	–
2	1	0	1	1	1	1	$\{1, 3, 4, 5\}$	12
3	1	1	1	1	0	1	$\{1, 3, 4\}$	6
4	1	1	1	1	1	1	$\{1, 3, 4\}$	6
5	0	1	1	1	1	1	$\{3, 4\}$	2
6	1	1	1	1	1	1	$\{3, 4\}$	2
7	1	1	0	1	1	1	*none*	0

Table 6.1: Several optimal configurations for one SBB-N module with one bit of neutrality. As more configurations are added, the number of associations that would be made by MACRO-H decreases due to conflicting allele co-occurrence. The 'associations possible' columns describe the pairings of 1-species that would be joined by MACRO-H if these example local optima were discovered (the count indicates the number of joins between all of these pairs). It takes very few examples to rule out all of the possible joins, despite the fact that a strong, if indecisive pattern is present.

One possible resolution is to reduce the strictness of the association rule. Instead of demanding that *all* evidence supports the join, we could make a join when *most* of the evidence provides support (with some threshold). However, committing fully to a permanent join based on incomplete data ('a soft rule for a hard join') is open to error. Instead, here we propose to make the strength of commitment proportional to the strength of evidence present.

6.2 Intermediate Relationships for Uncertain Covariance

6.2.1 Probabilistic Associations

While the example optima in Table 6.1 are sufficient to prevent the simple discrete rule from forming any associations, there is still a strong pattern present. By forming partial associations between species that co-occur with high-frequency we can make use of strong, but not unanimous correlations.

We define $X(i)$ as an estimate of the univariate probability of species i, (the proportion of demes where i is present). If two species are independent, we expect their bivariate frequency to be the product of the univariate frequencies:

$$B_{exp}(i,j) = X(i)\,X(j)\,. \tag{6.2}$$

A large deviance from this expected value indicates a correlation between i and j. We use this correlation to indicate the presence of an epistatic dependency. We define $B_{obs}(i,j)$ as the observed bivariate frequency of species i and species j (the proportion of demes where both i and j are observed).

We note that the value of B_{obs} has bounds that are functions of $X(i)$ and $X(j)$, such that $B_{min} \leq B_{exp} \leq B_{max}$:

$$B_{min}(i,j) = \max\left(0, X(i) + X(j) - 1\right), \tag{6.3}$$

$$B_{max}(i,j) = \min\left(X(i), X(j)\right). \tag{6.4}$$

Recall that an association $S_{i,j}$ is interpreted as a probability of species j co-varying with species i (see Table 5.2). Therefore, associations should only be formed when there is an indication of *positive* fitness interaction. We can make some simple and sensible restrictions on how to use the co-occurrence information. Firstly, we should not reinforce any associations when variables co-occur according to their univariate frequencies, as this indicates an ambivalent relationship. Secondly, when the deviance in co-occurrence is as strong as can be, we should make the symbiotic association as strong as possible. Thirdly, if the possible range of B_{obs} is zero, ($B_{max} - B_{min} = 0$), no information can be gained about the likelihood of interaction.

From the first and second statements above, we can define the end-points of the transformation: if $B_{obs} = B_{max}$ then set $S_{i,j} = 1$, and if $B_{obs} = B_{min}$ then set $S_{i,j} = 0$. There are many different ways that we could join up these two points, the simplest being a linear interpolation, as per $D(i,j)$ as defined in Equation 6.5. However, our preliminary experiments reveal that this rule leads to incorrect associations being formed too easily on account of spurious noise in the deme results.

$$D(i,j) = \frac{B_{obs}(i,j) - B_{exp}(i,j)}{B_{max}(i,j) - B_{exp}(i,j)}. \tag{6.5}$$

We note that Equation 6.5 is close to the χ^2 statistic, which is not quite suitable for our purposes. This is because the χ^2 aims to identify the dependence between variables, without specifying the direction of the relationship. We only want to reinforce *positive* relationships, and as such the directional information is vital. In this chapter we elect to use a lower threshold on the deviance information, respecting the condition that $S_{i,j} = 0$ if $B_{max} - B_{min} = 0$. Equation 6.6 describes this rule, where t is the threshold parameter

Table 6.2: Probabilistic Symbiotic Optimisation (MACRO-S): Main Procedure

1.	Allow d demes to run to their local attractor (see Table 5.1)
2.	Measure the co-occurrence between each pair of species within all deme attractors, and reinforce/update symbiotic associations (see Equation 6.6)
3.	Randomise each ecosystem composition and go to step 1.

below which all $S_{i,j}$ values are set to zero.[1]

$$S_{i,j} = \begin{cases} D(i,j) & \text{if } t \cdot D(i,j) < B_{obs}(i,j) < B_{max}(i,j), \\ 0 & \text{otherwise.} \end{cases} \tag{6.6}$$

There are several possible alternative modifications to the raw interpolation of the deviance metric (Equation 6.5), including raising to some exponent, logistic transforms, and step thresholds. However, the option described in Equation 6.6 is suitable for our purposes.

6.2.2 Putting Probabilistic Associations to Use

The full algorithm using probabilistic association updates follows the overall procedure described in Table 6.2, and associations are exploited using the procedure described in Table 5.2. The association updates defined in Equation 6.6 replace the symbiogenic joining rules from Table 5.3.

We name this variant of the MACRO framework as MACRO-S, where the ‘S’ stands for ‘soft joins’. In contrast to MACRO-H, there is no explicit dimensional reduction in this instantiation, even when associations are of maximal strength.

6.3 Behaviour on Modular Systems

To investigate the behaviour of the new algorithm in a principled manner, we consider the SBB-N problem described above. While we have described how the neutrality in the SBB-N is likely to cause difficulties for MACRO-H, it does have an explicitly modular structure. It is therefore the type of problem that we expect MACRO-S (a generalised version of MACRO-H) to be capable of efficiently solving, provided the generalisations are able to overcome these difficulties.

[1] Note that the association rules used in MACRO-H (Chapter 5) are functionally equivalent to setting $S_{i,j} = 1$ iff $B_{obs} = B_{max}$, and $S_{i,j} = 0$ else.

Neutrality, η	Discrete		Probabilistic	
(bits per module)	Successful	Evaluations	Successful	Evaluations
0	100/100	40087.0 ± 3.4	100/100	44771.25 ± 444.2
1	0/100	$> 10^7$	100/100	43491.26 ± 322.6
2	0/100	$> 10^7$	100/100	41965.67 ± 164.7

Table 6.3: Results of MACRO-H and MACRO-S applied to 400-bit SBB-N problem instances with various levels of neutrality, η. The success rates and mean evaluations to success are with respect to finding any configuration with globally optimal fitness (*i.e.*, any of the configurations in the all-1 plateau for all modules). For $\eta = 0$, this is the unmodified SBB, and the only instance that the discrete method can solve. The probabilistic implementation is able to handle the ambiguity in the local optima and solve all instances reliably. Note that the evaluation count for MACRO-S is more or less invariant across varying levels of neutrality.

6.3.1 The SBB-N Discriminates Discrete Joins from Probabilistic Associations

Initially we investigate the ability of both MACRO-H and MACRO-S on 400-bit SBB-N instances with $\eta = \{0, 1, 2\}$ bits of neutrality per module. Setting $\eta = 0$ provides a control, since this reduces the SBB-N to the SBB problem. We have not derived the exact number of demes necessary to reliably solve the problem with either algorithm, so we use a reasonably sized sample of $d = 50$. For MACRO-S, we set $t = 0.6$. We allow a maximum of 10^7 function evaluations for each algorithm For both symbiosis algorithms, the hill-climbing algorithm is described in Table 5.1: in each step, a single migrant group is introduced and tested. The focal migrant that seeds each migrant group is sampled without replacement. In MACRO-S, when associations develop of strength $0 < S_{i,j} < 1$, the migrant groups that form during subsequent hill climbing are not guaranteed to be identical each time; instead the symbioses bias the likely group constitution.

This experiment confirms that MACRO-H is prevented from making the associations necessary to decompose the problem and thus find a global optimum, as discussed in Section 6.1.2. With the number of demes set to $d = 50$, there is some contradiction preventing every association that would be made in the absence of neutrality. Without the correct associations, MACRO-H cannot reduce the effective dimensionality of the problem, and consequently the performance is not distinct from a non-adaptive restart-hill climber.

On the other hand, by using probabilistic associations, MACRO-S can handle the ambiguity and thus solve all instances reliably. The number of evaluations required is approximately the same for all values of η tested, reducing slightly with greater neutrality. This reduction is because constructing a migration group that has a high likelihood of at least 15 ones in a 20-bit module requires fewer/weaker associations than to create a migration group with at least 17 ones.

6.3.2 Probabilistic Rules are Robust to Large Sample Sizes

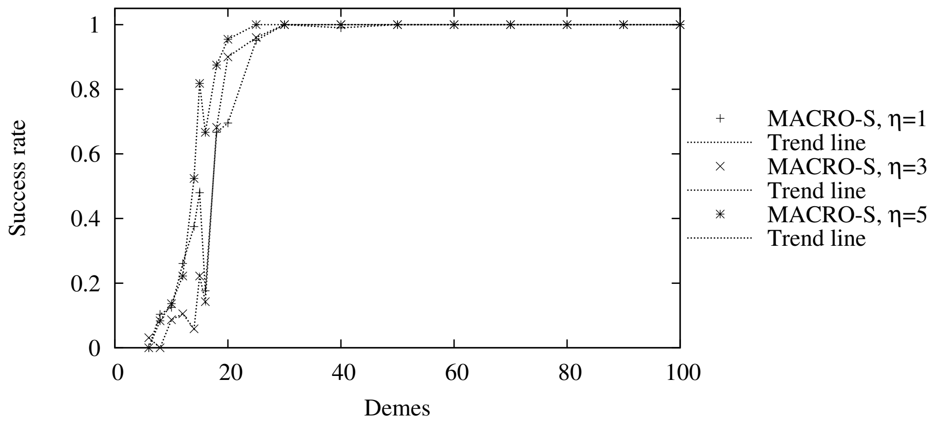
The locally optimal configurations on the all-1 plateau of the SBB-N are fairly consistent, but present some conflicts. As we saw in the previous experiment, when using a reasonable number of demes, these conflicts prevent MACRO-H from creating suitable associations. However, the rules (Table 5.3) are conservative enough that no incorrect associations are formed. The discrete method can actually solve this problem, if the number of demes is carefully chosen. That is, if few enough demes are used such that no contradiction is seen for a particular pair of species, a permanent association is made which reduces the dimensionality. As associations are gradually formed, MACRO-H is able to solve the problem. This gradual process takes more than two epochs (as would be the case if the dimensionality was reduced efficiently). Depending on a small number of demes is not a viable strategy: there is an inherent tension between having a sufficient sample size in order that module solutions are not missed (as considered in Section 5.4) and keeping the sample size small so as to avoid contradictions.

Here we investigate how sensitive MACRO-H is to this tradeoff, by varying the number of demes used in each algorithm for 400-bit SBB-N instances with $\eta = \{1, 3, 5\}$ per module. We observe the success rate of MACRO-H and MACRO-S over 100 runs, allowing a maximum of 10^7 evaluations. For MACRO-S, we set $t = 0.6$. The results are shown in Figure 6.2.

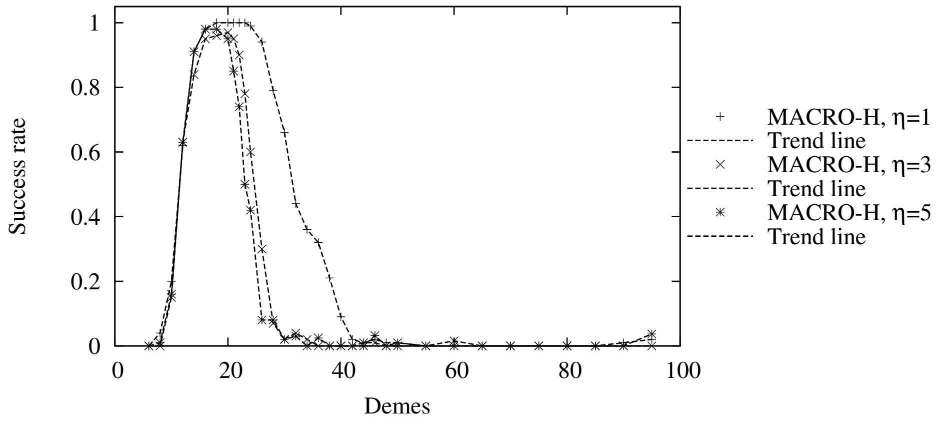
This experiment demonstrates that MACRO-H can only achieve high success rates for a narrow window of deme counts. The window is similar for each of the settings of η, becoming narrower with increasing neutrality. On the other hand, the accuracy of the information that MACRO-S uses to update association strengths is continually improved with an increasing deme count. Once a sufficient number of demes is used for MACRO-S to reliably find a global solution, the success rate remains at 1 for any further increase in deme count.

A further result is that even when MACRO-H has the number of demes carefully tuned to its peak in success rate, it is significantly hampered in terms of function evaluations. The ratio of evaluation count for the SBB-N to the SBB for this algorithm ranges from 4.8:1 ($\eta = 1$) to 20.0:1 ($\eta = 5$). Conversely, the evaluation count for the probabilistic algorithm is little affected by the level of neutrality.

In conclusion, we find that using probabilistic associations offer the symbiotic optimisation framework an increase in generality. Specifically, probabilistic associations are robust to this type of uncertainty, where forming appropriate discrete associations requires careful tuning.



(a)



(b)

Figure 6.2: Success rates of MACRO-H and MACRO-S on the SBB-N for varying numbers of demes. (a) Notice that using probabilistic associations can make better and better decisions with increasing deme count until it is successful all of the time. (b) Conversely, there is a narrow window where discrete associations can achieve a high success rate. With a large number of demes the uncertainty in co-occurrence relationships prevents this method from making appropriate associations and hence a global optimum is never found. Data points are connected to indicate the overall trends.

6.3.3 Probabilistic Associations on Unambiguous Problems

Here we investigate the performance of MACRO-S on the SBB. We aim to verify that the modifications in MACRO-S from MACRO-H do not detract from the ability to solve the unmodified provably difficult problems.

The parameter settings are as follows. The local search algorithm used in MACRO-S is as described in Table 5.1. The lower association cutoff is set to $t = 0.6$. Without any in-depth tuning of the deme count parameter, we use $d = 50$ for a range of problem sizes from $N = 49$ to $N = 576$, such that $k = Z = \sqrt{N}$. We plot the mean number of evaluations used to find the global optimum in Figure 6.3 (data from 100 independent repeats). We reproduce the results for MACRO-H and the equivalent non-associative local

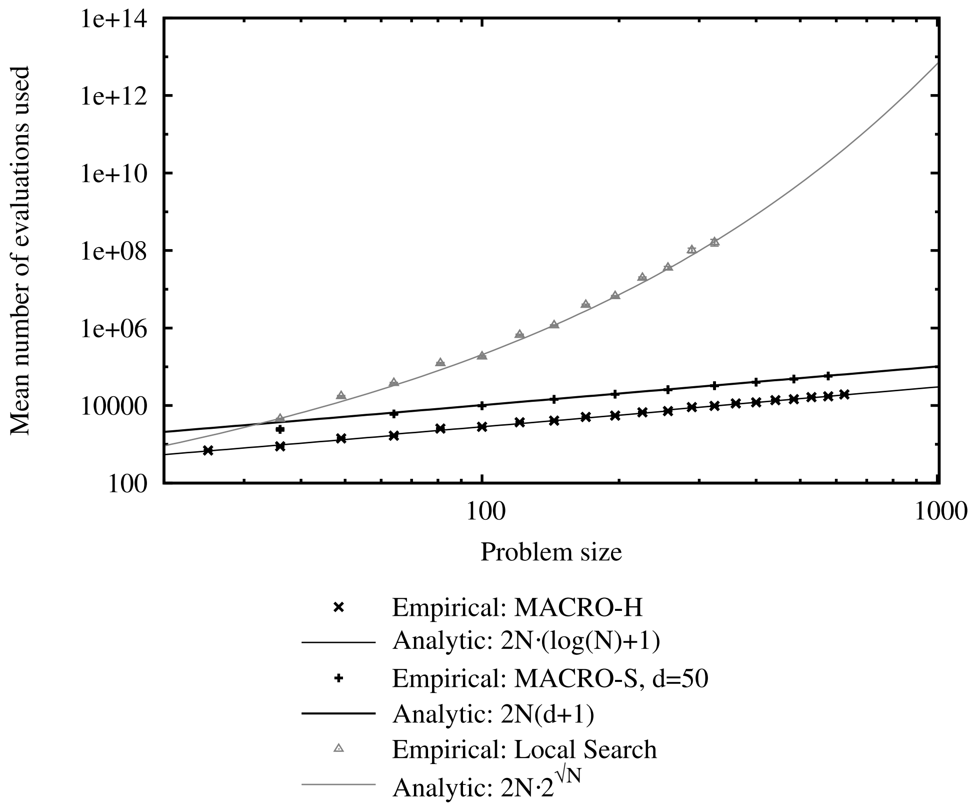


Figure 6.3: Number of evaluations taken to find the global optimum of the SBB problem, mean taken over 100 independent repeats. Both MACRO-H and MACRO-S are reported here, see Sections 5.4.2 and 6.3.3. A random-restart version of the local search mechanism that is a component of each of these implementations is also tested. Note the sub-quadratic time complexity of both symbiotic optimisation methods on this log-log plot. Data points are measured, while the lines plotted use analytical expressions, as given in the legend.

search algorithm from Figure 5.13 for comparison.

We find that $d = 50$ is a sufficiently large number of demes for MACRO-S to solve all instances in two epochs, as MACRO-H does. This is confirmed by how closely the empirical result is matched by the theoretical line of $\tau \cdot d = (2N + 1) \cdot 50$, where τ is the number of evaluations used by a deme. Although $d = 50$ cannot be an optimal value for all problem sizes tested, nor appropriate for all system sizes beyond this range, we aim here to demonstrate that using probabilistic associations does not detract from the ability to solve the SBB efficiently.

6.4 Behaviour on Systems without Explicit Modularity

In this section we examine a system that is not constructed from explicit modules, and investigate the advantage gained by MACRO-S over MACRO-H due to allowing associations of intermediate strengths.

6.4.1 Non-homogeneous Ising model on a 2D Lattice

In Section 6.3.2 we demonstrated how probabilistic associations can solve problems that exhibit ambiguity in the local optima. In this section we show how probabilistic associations can also be used to identify and exploit a lattice-based problem, in which the structure does not have such a straightforward decomposition.

The Ising model is a model of magnetism, in which sites (variables) can take two values: 'up' (+1) or 'down' (-1). Finding the lowest energy configuration, or *ground state*, is of physical interest, and can be formulated as a combinatorial optimisation problem.

We use a square two-dimensional lattice with 4-neighbour connectivity, periodic boundaries and zero external field. The bonds between sites $J_{ij} \geq 0$ are assigned random values. The energy function is given by:

$$H = \sum_{\langle ij \rangle} J_{ij} \sigma_i \sigma_j, \tag{6.7}$$

where $\langle ij \rangle$ denotes an edge in the lattice. The elements are similar in type but there is a small variation in the strength of each bond. The bond strengths are given by $J_{ij} = +1 \pm \phi$, where ϕ is a (static) noise term, randomly drawn from a uniform distribution, $\phi \in [0, 0.05)$.

A bond J_{ij} is satisfied when σ_i and σ_j are aligned. When all loci have variables that agree, all of the bonds are satisfied and hence give rise to the lowest energy state. In the Ising model an inversion of all states gives the same energy value, so there are two global optima, one at all-down and one at all-up. Note that while the dependencies in this problem are structured on a lattice, no information regarding the neighbours of any variable is given to any of the algorithms tested.

While investigating a simpler problem class would be preferable, our preliminary studies with a homogeneous Ising model (with all bonds $J = 1$) indicate that a simple hill-climbing method can solve the problem in approximately quadratic time. Since we use such a hill-climbing method as the initial basis of our algorithm, there is little scope for demonstrating a qualitatively improved ability through structural identification. Accordingly, we investigated the variant problem with a small amount of heterogeneity in the bond strengths as described above. After investigating the abilities of the MACRO algorithms, we learned that this problem formulation is very similar to the random bond Ising model introduced by Middleton (1995). This also uses all positive bonds with heterogeneous strengths, and hence creates a consistent problem. However, the strengths are drawn from $J_{ij} \in [0, 1)$ rather than $J_{ij} = +1 \pm \phi$. It remains as future work to investigate how important the different modes of randomising bond weights are.

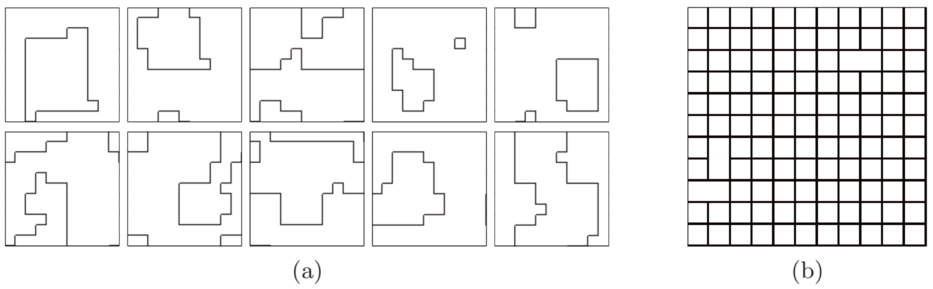


Figure 6.4: Domain boundaries discovered in several demes on an 11 × 11 non-homogeneous Ising model. (a) frames showing several examples. The boundaries are in different positions in each deme. (b) the superposition of domain boundaries from 30 demes. As the number of demes increases, the number of pairs of variables that cross a domain boundary in at least one deme increases. This significantly disrupts possible joins under the simple discrete rule set. In this example, all but three associations are ruled out.

6.4.2 Concordance Across Local Optima

This lattice-based problem does not leave the interactions in neat clusters with obvious boundaries, in contrast to the organisation of the SBB. One consequence of this is that local optima are distributed very widely across the space. In a global optimum, two neighbouring loci will have alleles that agree in value (*i.e.*, both 0 or both 1). However in a local optimum, some neighbours must have disagreeing alleles. A particular pair of loci will agree in some local optima and disagree in others. Even though neighbouring loci will agree more often than not in locally optimal configurations, an ensemble of local optima will present configurations in which particular neighbours disagree. Therefore, the association formation rules used by MACRO-H will not lead to associations that allow the problem to be solved efficiently. Figure 6.4 illustrates this issue (note that a domain is a cluster of contiguous variables that have the same spin value). The superposition of the domain boundaries from 30 demes shows how few associations a discrete rule can make.

While the discrete approach is almost entirely prevented from forming any associations, the inherent conservatism does not make incorrect associations either. This approach does gradually accumulate associations, as it did for the SBB-N. Each of these associations restricts the degrees of freedom in the system, and ultimately makes a global optimum accessible in the long run. Figure 6.5 shows the mean number of evaluations required to find a global optimum by each of our symbiotic optimisation algorithms. The problem size is set to $N = 225$. 100 independent repeats are performed for each value of d. In MACRO-S we set $t = 0.6$.

MACRO-S takes longer when it uses significantly more demes than necessary, as this linear increase in cost is unavoidable. MACRO-H suffers an increasing penalty with larger deme counts over and above the linear inefficiency of MACRO-S.

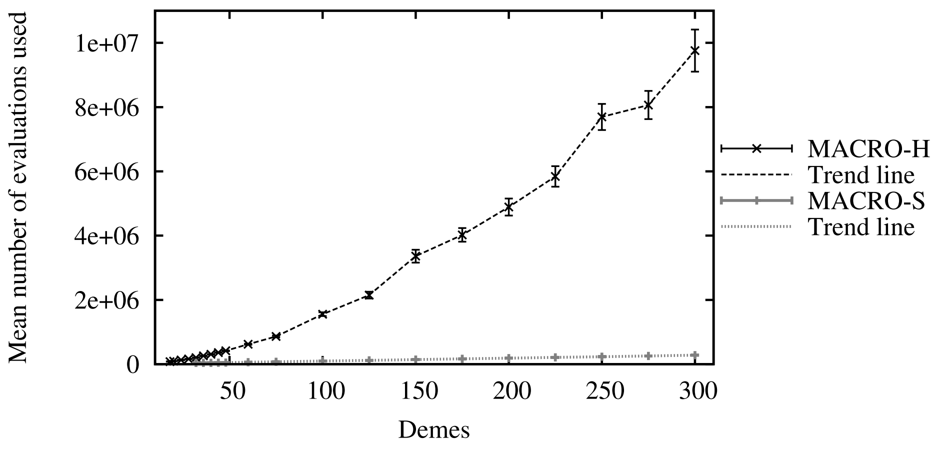


Figure 6.5: Evaluations required to find a ground state of a 15 × 15 non-homogeneous Ising model. The number of demes used has a significant impact on the discrete algorithm as more contradictions are found in the ensemble of local optima. The probabilistic algorithm can solve the problem reliably in a reasonable time even with a large number of demes. Data points are connected to indicate the overall trends.

6.4.3 Scalability

We test the performance of MACRO-H and MACRO-S across a range of problem sizes to assess their scalability. The symbiosis algorithms both use $d = 40$ for all problem sizes, and $t = 0.6$ for the probabilistic algorithm. We report the mean number of evaluations required to find a global optimum. For each problem size, all algorithms found a global optimum in each of 100 runs. The results are plotted in Figure 6.6, and include standard error bars but for both variants of MACRO these are negligible with respect to the markers. In addition to the experimental data for each algorithm, a line is fitted to the local search data to confirm the exponential relationship with problem size.

The probabilistic symbiotic relationships lead to the most efficient problem solving ability, indicating a sub-quadratic relationship with problem size. These results show that the multi-locus variation provided by utilising the symbiotic associations can result in the effective reduction in dimensionality of the search space. Moreover, forming appropriate associations does not depend on the search space having an explicitly modular structure.

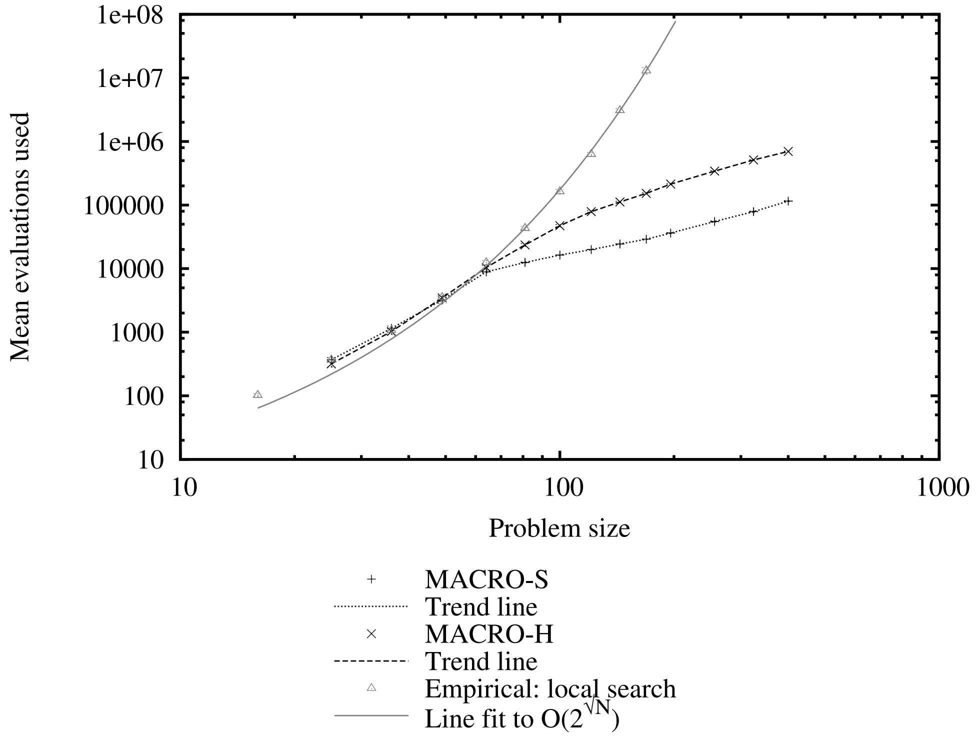


Figure 6.6: Both symbiotic optimisation methods have a sub-quadratic relationship with problem size, whereas the underlying local search method must explore an exponential number of basins before discovering one of the two globally optimal configurations. Note that for small problem sizes (under 64 variables), local search is sufficient to find the solution rapidly, and evaluation requirement for all three methods is similar. However once the number of basins expands to a moderate level, the symbiotic optimisation approaches offer qualitatively better performance. Data points for the MACRO algorithms are connected to reveal the overall trends, while we fit a line of the form $2^{\sqrt{N}}$ to the results of the local search algorithm.

6.5 Discussion

In this chapter we have demonstrated that the discrete join decision mechanism used by MACRO-H is obstructed when presented with patterns that contain uncertainty. The development of probabilistic associations, based on partially correlated contexts, can adequately handle this uncertainty. This advance allows us to avoid a situation where we are forced to make extreme decisions based on inconclusive information, without restricting progress.

The resolution has two components, each of which we demonstrate to be successful. First is the design of a mechanism that capitalises on correlations of intermediate strength. Second is the development of a corresponding mechanism that can suitably exploit the associations such that they inform the scale of optimisation appropriately.

The increase in generality has two underlying advantages:

1. Probabilistic associations do not have to fully commit to an irreversible union, immediately, or ever; and
2. This method retains information regarding correlations that are not sufficient to create a fully-committed join. Doing so makes progress in decomposing the problem, and avoids the need to start all over again (contrast the position of MACRO-H, which would take the same inaction of not making a join repeatedly even if presented with strong, but incomplete correlations in different epochs).

Each algorithm is robust to an 'overdose' of contexts from which to update associations, when those contexts are drawn from an unambiguous set, such as local optima in the SBB. The discrete join mechanism becomes brittle when the increase in contexts leads to a greater probability of a contradiction. But the probabilistic association mechanism remains robust when the estimated correlations from a sample of local optima tends towards that of the true distribution of correlations. This is advantageous since MACRO-S can effectively make use of any information provided to scale up the search space: even if the information is not strong enough to indicate a permanent union, it can inform an increased biasing of the correlations in future variation. The alternative of discarding such information is far less desirable.

We revealed the brittleness of MACRO-H by systematically introducing ambiguity to the local optima. This highlights in principle a type of environment where allowing intermediate strength associations is more general than only supporting extreme symbiogenic joins. A qualitatively similar ambiguity in the contexts used for association formation could be created if the separation in timescales between exploiting and adapting symbioses was reduced from that used in this chapter. We support the applicability of this specific principled test problem with the studies that employ the less idealised Ising model problem.

6.5.1 Related Work: Recursive Model Building from Local Optima, and Exploration in the Collapsed Space

We noted three papers in Section 5.1 that describe algorithms which perform dimensional reduction based on the results of some form of local search. The most closely related of the three is the building block hill climber (BBHC, Iclanzan and Dumitrescu, 2007), and a related extension (Iclanzan and Dumitrescu, 2008). The BBHC is an algorithm that runs a hill climber several times and records their final states in memory. It then explicitly collapses the search space when it finds pairs of variables to exhibit a bijection from the states stored in memory. Subsequent search is performed in the reduced space which allows it to solve building block problems that are hierarchically structured.

This algorithm is functionally close to MACRO-H, and obtains similar empirical results as to those in Section 5.4.4. However, there are differences to our discrete algorithm in the setup, and in our motivation. While we are interested in the applicability of concepts derived from the evolution of symbiotic associations in problem decomposition, Iclanzan and Dumitrescu are motivated by the integration of machine learning techniques to population-based search. These authors have taken a different direction in subsequent development of the concept. They consider problems constructed from building blocks that do not have useful gradients leading to the block optima – instead, considering deception and massive neutrality (Iclanzan and Dumitrescu, 2008). They show that in conjunction with a local search that uses *uncorrelated*, multi-locus mutations, using the local optima to collapse the search space can solve building block problems of this form.

We have developed our concept in a different direction by using probabilistic associations (in MACRO-S), which allows partial relationships between species. Moving away from a situation where an absolute and irreversible dimensional reduction is effected is a significant generalisation. The evidence we present in Sections 6.1.2 and 6.4.2 illustrates some of the limitations with pursuing such explicit dimensional reductions. Specifically, an approach that gets *less* likely to be able to proceed with a *better* sampling of the search space is unsatisfactory. However, this issue does not manifest itself in problems that are neatly defined with explicitly modular structures.

Houdayer and Martin (1999) present the genetic renormalisation algorithm (GRA), which builds on the concept of renormalisation groups to effect explicit dimensional reduction. As for the BBHC, a variable can only be part of one composite group. In addition, the GRA only collapses groups into higher-level variables when sites comprise contiguous blocks, which depends on knowing where each variable is located in the lattice *a priori*. Note that the BBHC does not apply such a restriction, nor use any *a priori* information regarding the system structure.

Mahdavi et al. (2003) apply a similar idea to automated module clustering in software. They identify common components from multiple local search runs to form modules

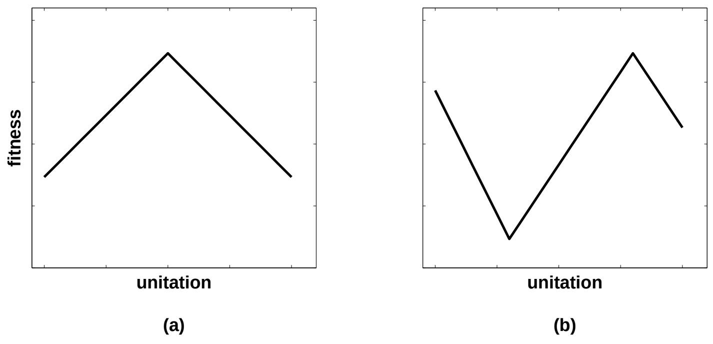


Figure 6.7: Two functions of unitation that are problematic for building pairwise models.

that are used as primitives in a second round of local search. Recursion is not explicitly mentioned but is straightforward in their framework.

6.5.2 On the Limitations of MACRO-S

In this chapter we have explored the use of evolving probabilistic associations for multi-locus variation, on systems with explicit modularity, and systems that do not have an explicit decomposition. Both of these system classes had some ambiguity in where local optima are, but there are recognisable correlations between variables. Therefore, it is possible to use symbiotic associations of intermediate strength to build an appropriate pairwise model that can faithfully represent the structure.

A different type of structure that would be problematic is when local optima are not ambiguous (unlike the SBB-N), but the class of local optima is best described by an order-k relationship, and there is a lack of consistent pairwise information in any representative sample of these local optima. Two examples are shown in Figure 6.7.

In (a), there is no systematic pairwise information that can build a good model; no correlations in the co-occurrence in any representative sample. However, because this function is unimodal, local search in the initial units would reliably find an optima. In (b), the covariant relationships will largely be accurate, but on average, the associations between each 1-species and the other 1-species are strong, so the difficulty would be in coordinating the right number of 1s with the few 0s necessary. This is contributed to by the fact that unitation functions treat all loci as interchangeable, or fungible (*sensu* Queller, 1997).

In both of these examples, the issue stems from the lack of a suitable pairwise model. Because most model-building EDAs aim to build (or start with) a pairwise model,[2] a function created from multiple modules with these properties is likely to be problematic

[2]except univariate EDAs such as PBIL, which do not build a model of any inter-variable dependencies at all – see Section 2.3.1 for a description of algorithms in this class.

for all EDAs. Coffin and Smith (2007b,a) highlight exactly this issue by showing that a multi-locus parity subfunction is a pathologically difficult problem for EDAs, since there is no low-order mutual information to build a model from across the high-fitness subset selected from random candidates.[3]

Two further cases where MACRO-S would also not be able to gain any traction include: 1) where the global optimum is unrelated to the other optima in the system; and 2) where the fittest module configuration has a vanishingly small basin of attraction, and that module cannot be decomposed further. These are also rather artificial limitations, and are likely to limit all stochastic search techniques.

6.6 Summary

In this chapter, we have demonstrated that using probabilistic associations offers a generalisation to the MACRO framework. In particular, this robustly handles situations where local optima at one scale are somewhat ambiguous, while retaining the ability to decompose structured search spaces. The probabilistic association formation mechanism used in MACRO-S can express a richer class of relationships than the joining rules used in MACRO-H. Moreover, probabilistic associations are arguably more natural since they can more closely match the information that the symbiosis matrix was constructed from. In this sense, probabilistic associations are less wasteful – information regarding species co-occurrences that are not at the extreme of the possible range are not discarded.

In the algorithms introduced in the previous two chapters (the RSSA and MACRO-H), once a symbiotic relationship is decided upon, the lower-level entities are permanently encapsulated into a single unit. This explicitly reduces the dimensionality of the search space, rendering optimisation more tractable. In contrast, MACRO-S does not perform dimensional reduction explicitly. Instead, the symbiotic relationships lead to dynamically created migration groups, whose distribution exhibits significant correlations between compatible species. This is a reduction in the effective degrees of freedom in the system that can act in a more general manner.

We have further demonstrated that separating the timescales upon which rapid exploration and adaptation in the decomposition occur is central to the ability of MACRO to automatically identify and subsequently exploit system structure. Furthermore, we have shown that the principles of MACRO apply to a broader class of systems, to include both explicitly decomposable, and lattice-based problem structures with ambiguity in local optima.

[3]For a 2-bit parity subfunction, the problem of coordination or anti-coordination is achievable with pairwise relationships, but beyond 2 bits cannot be represented.

Chapter 7

Distributing Symbiotic Association Formation

In the previous two chapters, we demonstrated the MACRO framework to be efficient at problem solving through its ability to automatically identify and subsequently exploit system structure. In this chapter, we aim to better understand the mechanisms behind the success of the MACRO framework. Simplifications to the association formation may allow us to refine the minimal requirements that can identify an appropriate decomposition to provide macro-scale search. Watson et al. (2009a) observe that the associative changes in MACRO-S obey a Hebbian protocol, which stimulates our investigation of MACRO in a neural substrate in this chapter.

Our goal in this chapter is to understand the essential mechanisms that afford the computational problem solving ability demonstrated in previous chapters. We are also motivated to understand whether our evolutionary-inspired approach can be plausibly interpreted as a biological model. To this end, the distributed nature of neural substrates may enhance the biological plausibility of MACRO, on account of each node dictating changes to its own state and associations. This situation contrasts with MACRO-H and MACRO-S, in which state and associative updates were made according to system-wide information.

Hebbian learning can provide changes to associations that are qualitatively equivalent to those implemented in MACRO-S, thus enabling a simpler, distributed implementation of MACRO. By testing this hypothesis, we enhance our understanding of the requirements for system decomposition mechanisms under the MACRO framework.

We develop a new MACRO algorithm based on a Hopfield network, and explicitly implement Hebbian learning to provide the associative changes. This algorithm applies both the variable state changes and adaptation of the decomposition in a distributed manner, using only localised feedback. We investigate how macro-scale variation is created in this algorithm.

We find that this implementation of the MACRO concept is capable of identifying structure in modular problems, and the dynamics of the network are drastically changed under the application of Hebbian learning. That is, the application of Hebbian learning to reinforce associations between components that co-occur at local optima in micro-scale search is capable of creating macro-scale variation that matches the system structure. In a problem that initially presents an exponential number of local optima, once the structure has been learned, the system only visits attractors with the highest overall utility. We also find that in less clearly structured environments, the evolution of associations can still lead to the discovery of significantly higher utility attractors than under a non-adaptive regime. In both cases, the modification to system dynamics occurs because of the ability of the system to generalise from the attractors that it has visited.

We demonstrate simplifications to the MACRO framework that further clarify the minimum requirements for this approach to provide effective problem decomposition and consequent problem solving. In particular, we show that applying Hebbian learning to reinforce correlations between variables that co-occur at local optima is capable of providing macro-scale search, if the learned connections are appropriately interpreted. The Hebbian learning mechanism needs only localised information to update associations between variables. Additionally, the learning can be applied gradually such that each local optima visited has a small influence on the correlations. These two facets are simplifications over MACRO-H and MACRO-S, which both perform a (simple) analysis of a batch of local optima to determine how to adapt the decomposition.

Overall, these results strengthen our understanding of what enables the MACRO approach to provide problem decomposition. The enabling mechanisms are: 1) a model-informed hill climbing process that can effect multi-locus changes according to a model of problem decomposition; 2) a method of positive feedback that reinforces the co-occurrence of variables in the system dynamics (Hebbian learning); and 3) a separation in the timescales of 1) and 2), such that association changes are performed mostly or wholly at local optima.

7.1 Credit Assignment

The work described in this chapter has been strongly influenced by successful results and processes described in previous chapters – in particular, the concepts central to the MACRO framework of using locally optimal contexts to inform changes in associations, and using those associations to modify the later behaviour of the system. However, this chapter's work also depends on the observation that successful relationships made in MACRO-H and MACRO-S form in accordance with Hebb's rule. This observation, and the subsequent development of a model to directly test this, was made by Richard Watson. I have been involved heavily in this research, but have not been the main driver. However, the contributions for which I claim credit are:

1. Conceptual development of these models, and algorithmic interpretation (partial);
2. Designing experiments that are able to illustrate and discriminate the abilities of MACRO-D, with respect to two controls HN-EI and HN-C (partial);
3. Investigation of properties of the RRB problem (total);
4. All figures and other results included in this chapter are produced directly from experiments that I executed (total);
5. Creating the text within this chapter (total).

I am not responsible for (RW=Richard Watson, CB=Chris Buckley):

1. The observation that Hebb's rule is implemented in MACRO-H (Watson et al., 2008, 2009a), and subsequent development of HN-EI and MACRO-D (RW)
2. Casting as a dynamical system (CB)
3. Moving from optimisation to dynamical systems consideration and language (CB)
4. The observation that energy minimisation on weights implements Hebb's rule (RW)
5. The work that shows the naturalisation of Hebb's rule (RW + CB)

Throughout this chapter, I have cited the following external work where I am not the primary initiator of a particular idea:

1. For the specific observation that reinforcing co-occurrence to encourage future co-occurrence is Hebbian: (Watson et al., 2008, 2009a)
2. For the design and method of HN-EI: (Watson et al., 2009a,b, 2010)

7.2 Hebbian Learning, Hopfield Networks, and Distributed Optimisation

Hebbian Learning

In Chapters 5 and 6 we identified mechanisms that led to the formation of associations that reflect the system structure. We achieved this through reinforcing the co-occurrence of commonly co-occurring variable configurations in an ensemble of local optima. Watson et al. (2009a) observe that reinforcing correlations in this manner is an implementation of Hebbian learning.

This chapter aims to directly test this observation, by explicitly implementing Hebb's rule (Hebb, 1949). We find it appropriate to study Hebbian learning in a neural substrate, and this allows us to draw on the literature in this field (*e.g.*, Linsker, 1988; McEliece et al., 1987; Gimenez-Martinez, 2000).

In particular, we use a Hopfield network (Hopfield, 1982) as an abstract representative for dynamical systems, and investigate the conditions under which it is able to perform global optimisation. Other work has used Hopfield networks for optimisation, including (Hopfield and Tank, 1985, 1986; Shih and Yang, 2002; Ackley et al., 1985). The manner in which we employ the Hopfield network is unique: we combine associative memory with energy minimisation to provide an optimisation technique. For a fuller review, see (Watson et al., 2009a).

Our method aims to reinforce components of states that have been visited frequently. While accepting that these states are in general local optima, we assume that nearly-decomposable problem spaces have components that are of high quality in multiple contexts. Therefore, we aim to allow such a component to be selectable for use in future search. This is in contrast to approaches that try to avoid locally optimal solutions that were deemed to be of poor quality (*e.g.*, Lourenco et al., 2003; Li and Goodman, 2007; Frost and Dechter, 1994).

In explicitly using Hebb's rule in a network model, we move away from analysing a batch of local optima together. Hebbian learning allows us to recognise that reinforcing node co-activation gradually in each visited local optima *could* have the same result as the batch mode association formation (since it has a very similar underlying motivation). The experiments in this chapter demonstrate conditions where these two modes are qualitatively equivalent.

Hopfield Networks

There are several different forms of Hopfield networks, which include varying the type of states (continuous or discrete), and how the dynamics are updated (system relaxation according to gradient descent on the energy function, or perturbing the system to a new state and accepting the change if it is lower energy). See (Watson et al., 2009a) for further detail on the distinction between these forms.

We find it most suitable for our purposes to employ discrete states with the 'Monte Carlo' method of running dynamics, for two main reasons. Firstly, this thesis has largely focused on combinatorial optimisation using black box optimisation assumptions, and the Monte Carlo approach is compatible with the trajectory optimisation methods that we have encountered.[1] Secondly, central to the MACRO concept is the evolution of correlated multi-state changes, which should be accepted or rejected as a unit according to the improvement in utility. The Monte Carlo approach is a straightforward way to execute this operation.

Distributed Optimisation

There are various motivations behind research in distributed system control. These include understanding how complex organisation emerges in ecosystems (Levin, 1998) and in the economy (Arthur et al., 1997; Daniels et al., 2003). Further, there are many applications in which localised decision making is favourable, particularly where costs of communication are high (*e.g.*, in satellite systems, van der Horst et al., 2009), and where robustness to failure and adaptability are desirable (Vinyals et al., forthcoming).

There are many evolutionary-inspired optimisation techniques that can feasibly be implemented in a distributed substrate. These include the traditional genetic algorithm, as well as ant colony optimisation. A number of studies consider a competitive coevolutionary approach to problem solving, where one population represents sample problem instances, the other population represents solutions (Hillis, 1990; Pagie and Hogeweg, 1998; de Boer and Hogeweg, 2007). The tasks evolve to provide difficult patterns for the solution population to cover, rather than directly being tested on the full, general set of tasks. Each of the populations uses a localised fitness evaluation, and overall, these algorithms are distributed in implementation. It is not obvious that an EDA could be distributed, since the model building stage is often very complicated, and can require an auxiliary search to identify a well-fitting model. While algorithms such as the compact GA (Harik et al., 1999) could be distributed, it is a univariate model building algorithm, and as such cannot provide any associative learning.

[1]This mode requires a weaker assumption regarding the system that we want to optimise. If a function is not differentiable, or more generally, its form is not known, we can separate the system into a learned matrix and an external oracle function (Watson et al., 2009a).

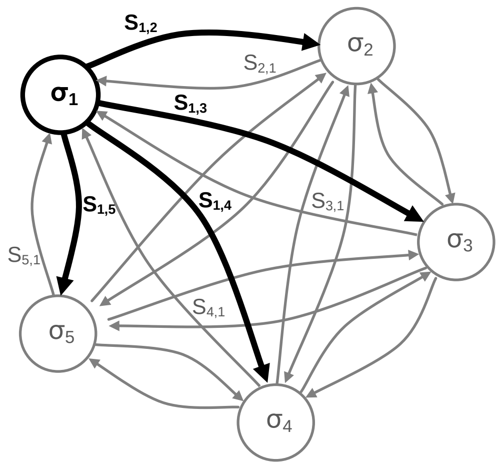


Figure 7.1: Representation of the MACRO-D algorithm: the system comprises nodes that can take on binary states, and weighted edges that describe the association strengths between nodes. Each node is responsible for both updating and storing its own state, as well as the outgoing association strengths. Note however that nodes have no direct control over the strength of incoming edges. These properties mean that the system is implemented in a distributed fashion, with no information on associations or node states stored centrally. In this example, node σ_1 is highlighted in bold, with all of the parts of the system for which σ_1 has direct control. The incoming edges for σ_1 are of interest to the groups that σ_1 may become a member of, but are governed by nodes σ_2–σ_5 and not σ_1 itself.

7.3 The Distributed MACRO Algorithm

Here we describe MACRO-D, a distributed association formation algorithm that is based on the same principles as MACRO-S. Our new algorithm is based on a Hopfield network, and as discussed above, we use a discrete state, Monte Carlo state transition version (Hopfield and Tank, 1985). The Hopfield network has N nodes, $\boldsymbol{\sigma} = \sigma_0, \sigma_1, \ldots, \sigma_{N-1}$, and each node can take states $\sigma \in \{-1, +1\}$. Each network node represents a problem variable. In our algorithm edges between nodes are used to represent the associations between the nodes that control how multi-node groups are formed. Figure 7.1 provides a schematic of the algorithm representation.

The main procedure is in Table 7.1, and the association-informed system dynamics are described in Table 7.2. Note that step 2(a) uses selection with replacement. This is in contrast to MACRO-S, which uses selection without replacement. We make this change in sampling regime because selection with replacement does not need a list to be maintained centrally. Association strengths are changed according to Equation 7.1, and as in MACRO-S, we only modify associations when the system is at a local attractor

Table 7.1: Hopfield-Based Associative Evolution Model: Main Procedure

```
1. Randomise the state on all nodes $\sigma_i \in \{-1, +1\}, \forall i$
2. Relax the system to its local attractor (see Table 7.2)
3. Reinforce each association strength according to Equation 7.1
4. Go to step 1.
```

Table 7.2: Association-Informed Hopfield System Dynamics

```
1. Evaluate the initial composition ($\rightarrow E_p$)
2. Form a perturbation group $g$:
    (a) Randomly select a node index, $m$, and flip the state $\sigma_m$ in $g$
    (b) For each node with index $x \neq m$
        i. With probability $|S_{m,x}|$, assert a new state on $\sigma_x$:
              Set the state of $\sigma_x$ in $g$ as $\sigma_m \cdot \mathrm{sgn}(S_{m,x})$
3. Temporarily introduce $g$ to the system, i.e., all states defined in $g$ are asserted
   onto the system
4. Evaluate the modified system configuration ($\rightarrow E_m$)
5. If $E_m \leq E_p$, allow the migrant group $g$ to remain permanently, and set $E_p \leftarrow E_m$
6. If maximum evaluation count not reached, go to step 2).
```

(optimum).[2]

$$S'_{ij} = S_{ij} + \lambda \sigma_i \sigma_j. \tag{7.1}$$

That is, updating the association strength from node indices i to j only depends on the states σ_i, σ_j, and the previous association strength $S_{i,j}$, which is maintained by node index i.

Note that all association strengths are initialised to zero, and in this case, only one state is updated at once. Once associations have evolved away from zero, nodes can form groups where other members adopt states that are compliant with themselves. When this is the case, the states of multiple nodes change at once. This is in keeping with MACRO-S and MACRO-H. We denote the learning rate for MACRO-D as λ_S.

Fitness Evaluations This restriction on distributed fitness evaluation when using black box optimisation assumptions may appear unusual. However, because we are using fitness functions that can be described as a sum of pairwise dependencies, and the interdependencies between the nodes that define the energy function are symmetric, we can distribute the fitness evaluation and it is equivalent to individual changes.

[2]To have a system that was truly autonomous, we could equivalently modify associations according to co-occurrence on every timestep, with an appropriately reduced rate, $\approx \lambda/T$. However, this is computationally more expensive and, as an optimiser, we specify the timescale T between perturbations so the update could also be specified on the nodes. We use an approximation that assumes system dynamics to be dominated by the steady state, and not the transient – which is achieved with an appropriate separation of timescales.

Contrasting MACRO-S and MACRO-D

The components that both MACRO-S and MACRO-D have in common are central to defining the MACRO framework. These include:

- Initially, the local dynamics provide micro-scale search with single locus changes, and once symbiotic associations have evolved, the dynamics afford macro-scale variation with correlated multi-locus changes.
- The timescale upon which adaptation occurs in association strengths is separated from the faster changes in system configuration.
- Associations are reinforced between species (MACRO-S) / state configurations (MACRO-D) according to their co-occurrence at local optima/attractors.

However, there are also differences. Firstly, MACRO-S uses an ensemble of demes, MACRO-D considers a series of single patterns to gradually reinforce associations. Secondly, while MACRO-D explicitly applies Hebbian learning at each local optimum visited, MACRO-S reinforces associations between variables that co-occur in high frequency in local optima, which is an implicit form of Hebbian learning. We demonstrate that neither difference is important in successfully optimising nearly decomposable problems. Three important differences include:

- In MACRO-S, the ensemble could establish a baseline expected co-occurrence frequency – but visiting a single pattern at once cannot recreate such a baseline. Instead, MACRO-D simply reinforces associations between components that are co-activated at local optima.
- Furthermore, the changes in association strengths are governed by each node, rather than using a system-wide decision.
- Finally, the primitive units in MACRO-H and MACRO-S represent single values, whereas in MACRO-D, the primitive unit represents a variable. We discuss this in greater depth below.

Moving to the Hopfield network substrate from our ecosystem substrate requires an implementational shift. Where in MACRO-H and MACRO-S the atomic units represent a single value (one node state), in MACRO-D the atomic unit is a node that can take on multiple values (all of the node states in the alphabet). Using nodes is a stronger assumption, but one that is in line with many studies that use networks to model ecosystems and their dynamics (*e.g.*, Poderoso and Fontanari, 2007; Solé et al., 2000; Martín González et al., 2009) and other complex systems more generally (*e.g.*, Sun and Deem, 2007; Watts and Strogatz, 1998; Ciliberti et al., 2007; Newman and Girvan, 2004). The main impact that moving from representing values (with species) to representing variables (with nodes) has is on group formation:

- A perturbation group cannot be overspecified (*i.e.*, defining multiple values for the same node) since a nonzero association dictates either one value or another for a particular locus. In MACRO-S it is logically possible to specify nonzero associations for more than one species that occupies the same niche. The case in MACRO-D is simpler: any node can only specify a preference for one of the values at another particular niche. In both models, perturbation groups are underspecified in general (*i.e.*, defining values for fewer than N nodes).

- The motivation for a strong association in the MACRO framework is that a node is able to specify (or encourage the likelihood of) a partial environment in which many of its dependencies are satisfied. In order to facilitate this in MACRO-D, associations are interpreted in two parts. The magnitude controls the likelihood that an association will have an influence. The sign controls the alignment of the second node with respect to the focal node.
  With this procedure, it is possible that by satisfying the dependencies of node σ_i, other dependencies may be violated, causing the overall system utility to be decreased.[3] However, our rationale is to allow the creation of such perturbation groups, even if the group will ultimately be rejected.

7.3.1 Control Algorithms

To investigate the system behaviour when Hebbian learning is used to form associations, we consider two control algorithms. As for Chapter 6, we compare with an algorithm that is equivalent except that it cannot modify associations. Such a comparison is important to verify that the evolvable associations provide a non-trivial result. This algorithm is equivalent to a single-bit-flip hill climber, which samples with replacement. We refer to this as the non-adaptive control algorithm, or HN-C, and we implement it as per Table 7.3, with step 3 ignored.

As described, our motivation for designing MACRO-D follows from the observation that the association formation in MACRO-S effectively implements Hebbian learning. For this purpose, a Hopfield network is a suitable substrate in which to develop an implementation of MACRO. However, there are other methods for integrating Hebbian learning into a Hopfield network used for optimisation. A straightforward method is to apply Hebbian learning to the weights of the system, rather than to define macro-scale variation. We can assess the importance of the method of interpreting the evolved associations using such a control algorithm.

Specifically, we use an algorithm from prior work (Watson et al., 2009a,b). This algorithm (HN-EI) also uses Hebbian learning to change association strengths between node

[3]Some of the dependencies of σ_i may be unsatisfied overall given the current context, if it has not fully specified all of its symbionts. Thus, it is possible to specify correlations that are ultimately rejected, and such a decision is made using only the focal node utility u_i.

Table 7.3: Hopfield-Based Interaction Evolution Model (HN-EI)

```
Set the learned weight matrix to the problem matrix: $\mathbf{W}^L = \mathbf{W}$
While exit conditions not met,
1.  Randomise the state on all nodes $\sigma_i \in \{-1, +1\}, \forall i$
2.  Relax the system to its local attractor $\boldsymbol{\sigma}^\star$: for $\tau$ timesteps,
        (a) Select, with replacement, a random node $\sigma_m$.
        (b) Measure the individual utility of $\sigma_m$: $u_m = \sum_j^N \sigma_m w_{mj}^L \sigma_j$
        (c) Flip the state of $\sigma_m$ to $\sigma'_m$, and measure its utility: $u'_m = \sum_j^N \sigma'_m w_{mj}^L \sigma_j$
        (d) Accept state change if $u'_m \geq u_m$
3.  Modify weight matrix by applying a Hebbian update, according to Equation 7.2
4.  Go to step 1.
```

state combinations found at local optima. However, where it differs from MACRO is that nonzero associations are interpreted as a modification of the utility function, representing a change in the quality of an interaction when it happens. This contrasts with MACRO-D, which modifies the likelihood of a particular interaction, but not the quality of such an interaction when it happens. The important functional difference is that MACRO-D evolves correlated multi-locus variation, and HN-EI does not.

Procedurally, the algorithm for HN-EI is described in Table 7.3. We denote the learning rate for HN-EI as λ_{EI}. The algorithm starts with an initial weight matrix $\mathbf{W}$, which defines the problem. As Hebbian learning is applied, this matrix is modified. We denote the modified weight matrix as $\mathbf{W}^L$, to distinguish it from the original weights. We update the system states according to individual utility calculated using the evolved weight matrix. However, for a meaningful comparison with other algorithms, we report the system utility according to the original system weights. Changes to the weight matrix are made according to Equation 7.2, where τ is the number of timesteps in an epoch. This is an implementation of Hebb's rule (Hebb, 1949), and in (Watson et al., 2009b) we show that these Hebbian changes are in the direction that will maximise the utility of an individual node. We cap all weights to have a maximum magnitude of 1.

$$w_{ij}^L(\tau + 1) = w_{ij}^L(\tau) + \sigma_i(\tau)\,\sigma_j(\tau)\,\lambda_{EI}. \tag{7.2}$$

These changes will act to reinforce the likelihood of a particular state re-occurring. As we discussed in Chapters 5 and 6, identifying epistatic dependencies in arbitrary contexts is a significant challenge. In those previous chapters, we reinforce common components present at local attractors, and in HN-EI this same protocol is observed. That is, we use the same idealisation for HN-EI as for MACRO-D that changes to interaction coefficients are only made at the end of each trajectory.

7.4 Experiments

We perform three sets of experiments to investigate the system behaviours. Specifically, we consider the optimisation ability in a nearly-decomposable system (1) and in a more general class of problems defined with a matrix of random values sparsely distributed (3); and modifications to the system behaviour as associations evolve (2).

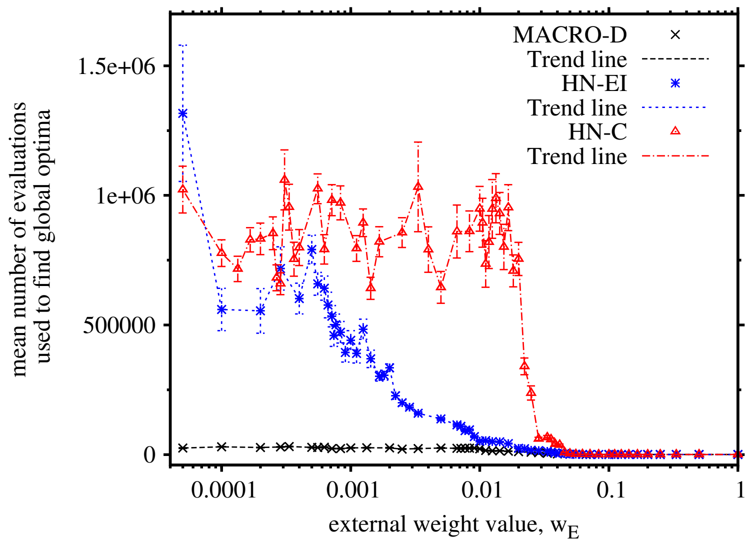
The number of timesteps in an epoch, τ is set to $\lceil \frac{3}{4} \cdot T \rceil$, where $T = eN \ln N$ for all three algorithms. T is from the result of Mühlenbein (1992) that $T = e^{\mu} N \ln N$ is the expected time for a mutation algorithm to solve *onemax*, where μ is the expected number of mutations per timestep (initially for our local search mechanisms in this chapter, $\mu = 1$), and N is the number of variables in the problem. By inspection, we find that the system has reached an attractor within $0.75 \cdot T$ in almost all cases (see also Section 5.3.7). The learning rate for HN-EI and MACRO-D are tuned for each experiment to reveal a strong effect with respect to the non-adaptive HN-C algorithm. Values tested are within the range $\lambda_S, \lambda_{EI} \in \left(2 \times 10^{-5}, 0.1\right)$.

7.4.1 Experiment 1: Optimisation Ability on the VSM

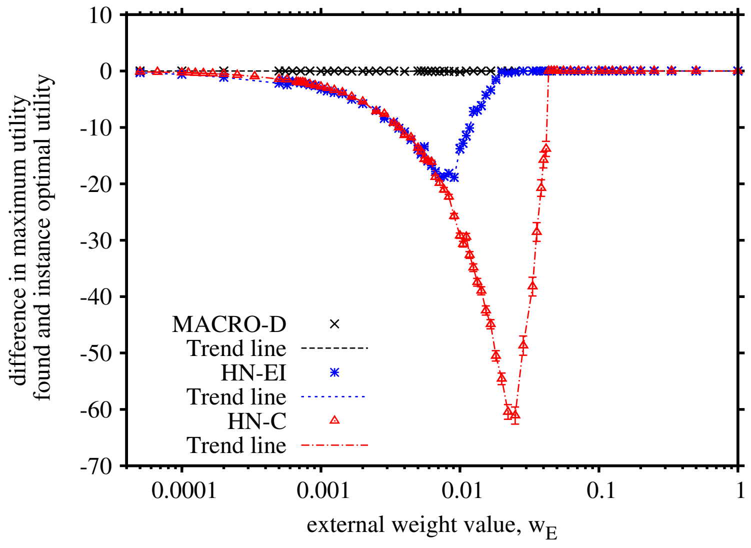
Here we investigate the optimisation ability of MACRO-D on the VSM (introduced in Chapter 3). We have previously established this form of problem to be exponentially difficult for local search, but in principle is solvable by methods that can produce specific multi-locus variation that corresponds to the modular structure. We consider instances with k=5, Z=20, $N = Zk = 100$, such that when the weight ratio $w_I : w_E$ is sufficiently high, there are $2^{20} \approx 10^6$ local optima, only two of which are globally optimal. We examine the ability across a wide range of weight ratios, and compare with HN-EI and HN-C. We set w_I=1 in all experiments, and report the value of w_E. We perform two sets of experiments, (a), to find the number of timesteps (evaluations) required to find a global optimum; and (b), to find the quality of optima discovered after a fixed period.

The results for the evaluations required to find a global optimum are given in Figure 7.2 (a). For all three algorithms, $\tau = \lceil 0.75T \rceil = 939$. The learning rates must be selected for HN-EI and MACRO-D. We choose these to provide the fastest optimum, given a success rate of at least 99% over 100 independent runs for each instance, and subject to the learning rate monotonically decreasing as the weight ratio decreases.

In the second experiment, we measure the highest utility found by each algorithm after approximately 50,000 function evaluations, in 100 independent repeats. The specific upper evaluation budget is set to $\lceil \frac{50{,}000}{939} \rceil = 54$ restarts. These results are given in Figure 7.2 (b). The learning rates for HN-EI and MACRO-D are selected to maximise the mean highest utility, subject to the same conditions for experiment (a).



(a)



(b)

Figure 7.2: Performance of MACRO-D (plus controls) on the VSM. (a) number of timesteps required to find global optimum; (b) maximum utility found after ≈50,000 timesteps. In both graphs, data points are connected to facilitate visual grouping of each data set.

Time to Find a Global Optimum

This experiment (Figure 7.2 (a)) can help us to understand when the performance between different algorithms is discriminated, and thereby understand the properties that each algorithm is able to exploit. Let us first consider the behaviour of HN-C, which reveals the difficulty of the problem for algorithms that can only vary at the micro-scale.

When the weight ratio is high, few or no local optima are present (see Appendix A), and HN-C can rapidly find a global optimum – for ratios higher than 14:1, this takes single epoch on average. At the weight ratio for which all attractors become present, HN-C rapidly transitions to taking approximately 10^6 evaluations to find the solution (with a broad distribution) for all lower ratios.

For intermediate ratios, below where HN-C finds the problem easy, the time that HN-EI takes to solve the problem stays lower than for HN-C for much of the range tested. This indicates that HN-EI can still generalise across the attractors that it visits (Watson et al., 2009a), providing a qualitative improvement over the local search mechanisms that underlies the algorithm. However, as the weight ratio decreases, the strength of the higher-scale signal (from between-module dependencies) weakens to such a point that many basins must be sampled to correctly reinforce the structure. By around $w_I : w_E$=2500:1 (0.0004), the necessary learning rate is so low that the number of initial conditions sampled tends to the number that is sufficient for local search to find a global optimum. Consequently, at this end of the range, the behaviour of HN-EI is not qualitatively distinct from HN-C.

The approach taken by MACRO-D is distinct from both HN-EI and HN-C: the evolution of associations that inform macro-scale variation that can reflect the modular structure of the problem. Even when the gradient that indicates which combinations of module solutions are compatible is weak, MACRO-D can follow that gradient. Provided that there is *some* signal, MACRO-D solves the problem very quickly – and this is the case whenever w_E is above zero.

Trade-off Between Difficulty and Quality of Attractors

Figure 7.2 (b) shows the highest quality of attractor that is discovered in a reasonable timescale by each of the algorithms. MACRO-D finds the global optimum in all cases, which corresponds to the result in Figure 7.2 (a) – where it solves the problem in approximately 30,000 evaluations or less.

HN-C shows the largest deviation from the globally optimal utility, for intermediate weight ratios. At high ratios, the absence of local optima means that HN-C reliably finds a global optimum. The lowest utility is at 45:1 (0.0222), and below this ratio, the relative utility discovered gets closer to optimal. This is not because there is a reduction in the number of optima, but because the range in quality of local optima is reduced when the external weights are not as significant. We discussed this trade-off in Section 3.5. Therefore, when the ratio has reached 10000:1 (0.0001), although HN-C does not find either of the particular configurations that are globally optimal (as Figure 7.2 (a) demonstrates), the penalty in utility is only 0.1%.

Importantly, there is a region for which HN-EI outperforms HN-C, below ratios sufficient to create many local optima, but above ratios where the quality of an average solution tends to that of the globally optimal solution (Watson et al., 2009b). That is, there exist instances that are both difficult and worth solving. Furthermore, in this region of problem instances, MACRO-D significantly outperforms HN-EI.

7.4.2 Experiment 2: Change in System Dynamics on the VSM

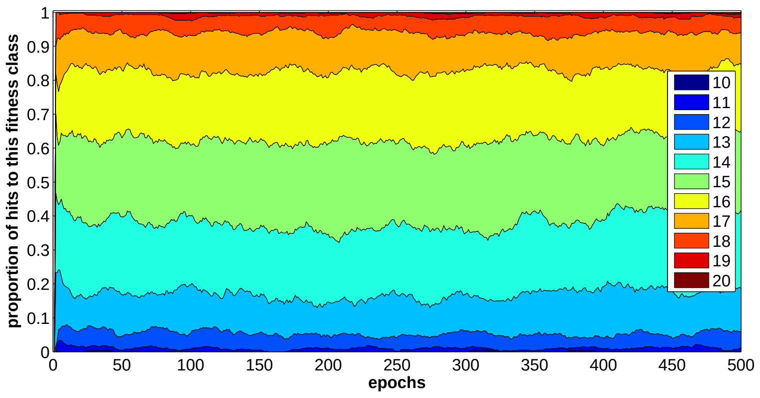
We are interested in understanding how the system dynamics change over time as a result of the learning. We find that the VSM is a suitable system to investigate this, since we know enough of the system properties to measure a meaningful result. In particular, we know the number and location of local optima, as well as their utility. For suitable weight ratios, the optima fall into $Z/2+1$ utility levels, and the aggregate size of basin of attraction for each utility level is given by the binomial distribution, according to the number of modules that are satisfied in the same direction. There are two global optima in a total of 2^Z optima, and for $w_I : w_E \rightarrow 1:0$, each optima have the same sized basin (*i.e.*, the chance of finding a global optimum from a random initial condition is $1/2^{Z-1}$). When there are non-negligible external weights, the basins of the higher utility optima are increased. Thus, finding a global optimum from a random initial condition is more likely than $1/2^{Z-1}$ but still very small.

Therefore, we investigate the dynamics of the system by measuring the proportion of epochs in which each of these attractor classes is visited, and observe its behaviour as associations (MACRO-D) or interactions (HN-EI) are evolved.

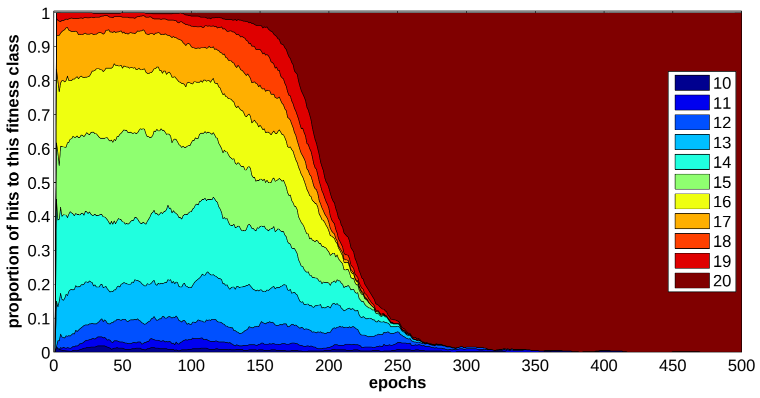
Each algorithm visits many states during its trajectory in each epoch, and we assume that the timescale used is sufficient for the final state to be locally optimal. In HN-EI and MACRO-D, associations are reinforced at this point. We also measure the total system utility here. We run each algorithm for 30 independent repeats, and report the distribution of attractor quality across all repeats. We apply a window of 20 epochs over the data to obtain a general trend. Note that the behaviour of HN-C within one epoch is unrelated to the behaviour in previous epochs (*i.e.*, it features no form of learning) and therefore there should be no meaningful trend in the optima visited over time; we simply use this result as a comparison of default behaviour. As in experiment 1, we measure the system utility in the original function for all algorithms.

We set the VSM parameters to k=5, Z=20, N=Zk=100, and $w_I : w_E$=100:1 (this weight ratio gives a difficult problem for local search, with a non-negligible large-scale signal for HN-EI – see experiment 1 (b)). We set the learning rates to λ_S=0.001, λ_{EI}=0.0015. For all three algorithms, $\tau = \lceil 0.75T \rceil = 939$. The distribution of attractor qualities are shown in Figure 7.3.

(a)



(b)



(c)

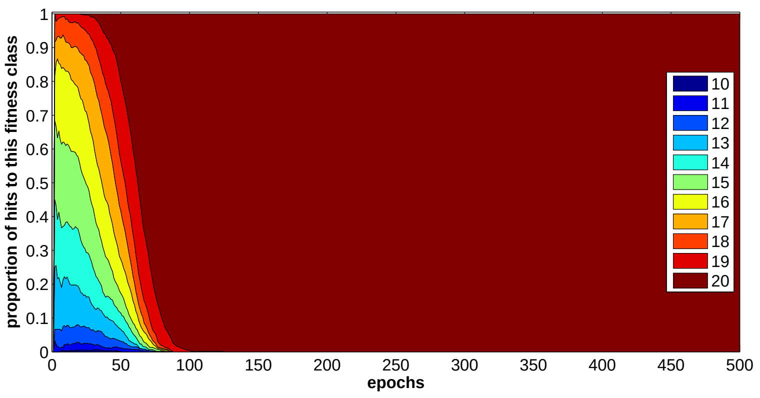


Figure 7.3: How the distribution of basins of attraction visited changes as (b) interactions (HN-EI) and (c) associations (MACRO-D) are evolved, in the VSM. Contrast these evolved dynamics with the non-adaptive control HN-C in (a).

Associative Evolution Causes a Drastic Change in System Response

Figure 7.3 (a) shows that without a mechanism to learn the structure, the system behaviour does not meaningfully change over time, and only very rarely are the highest utility classes visited. On the timescale for which this experiment was performed, a globally optimal configuration was visited only very rarely (17 of 15,000 epochs). This matches with the result in experiment 1(a) where the mean time to find a global optimum was 1010.0 epochs.

Conversely, the learning mechanisms in both HN-EI and MACRO-D have a significant impact on the quality of attractor visited. HN-EI is capable of enlarging the basin of attraction of the highest utility attractor, such that even before it has been visited by HN-C, HN-EI visits it from all initial conditions by approximately 350 epochs (Watson et al., 2009a). The response of MACRO-D is faster than that of HN-EI, having successfully modified the dynamics to only visit global optima from all initial conditions within approximately 100 epochs. The associations that form from the micro-scale dynamics make variation at the level of module solutions very likely. This macro-scale variation allows MACRO-D to explore module combinations to efficiently traverse the landscape, and consequently a global optimum is discovered from any initial condition.

This experiment illustrates that the system behaviour of both adaptive algorithms is modified to more reliably discover the highest utility state, which was not readily available from experiment 1. It is not just that the algorithms provide a method to find these configurations, and then recall it. Rather, the dynamics have been modified to make the discovery of these configurations much more likely even if the system is perturbed. By considering the algorithms as dynamical systems in addition to as optimisers, we learn more about the robustness of the result. As discussed in (Watson et al., 2009b), the timescales within which HN-EI and MACRO-D have modified the system such that a global optimum is the only attractor in the system are shorter than the mean number of epochs used by HN-C to hit a global optimum for the first time. This supports the argument that both HN-EI and MACRO-D generalise over the components that they have been exposed to in order to *predict* where higher utility optima are, and that the size of the basin of the global optima has been increased before it is visited.

7.4.3 Experiment 3: Optimisation Ability in Random Parameterisable Modular Problems

Here we investigate the generality of MACRO-D by considering the performance on an ensemble of randomised utility functions created by sparse matrices. We investigate a range of problem structuring, from a matrix that has a sparse number of nonzero dependencies with uniformly random distribution, through to the same density of dependencies organised on the block diagonal. We refer to this class of problem matrix as *rewired random-block* problems (RRB).

Where nonzero, the dependencies take on weights in the set $\{-1, +1\}$, and therefore open up the possibility of neutrality and inconsistency. We define the 'rewiring' scale r in a similar manner to the structural modularity scale for the VSM (see Chapter 3). For $r = 0$, nonzero values are randomly placed according to Equation 3.4. This creates

Z blocks of k nodes, each separable (*i.e.*, with no between-block dependencies). For $r = 1$, the maximal amount of rewiring gives a distribution equivalent to a uniform random placement of nonzero weight values. For intermediate rewiring values, some nonzero dependencies are moved from the block diagonal to a new random location outside that region, following the same protocol as for rewiring internal weights in the VSM (described in Section 3.3).

The VSM problem, used in the previous two experiments, is valuable because we know many properties regarding structure, position of local optima, and it can be intractable for local search. However, it is clearly idealised in some respects such that these properties are accessible. The RRB problem allows us to consider a more general formulation where structures are less apparent (particularly as $r \rightarrow 1$), and not all of the dependencies are mutually satisfiable. The price that we pay for this generality is weaker knowledge on statistics of the problem, including the knowledge of where the global optima is, or its utility. This changes what we can report.

Because we do not know where the global optima is, we allow a period of adaptation p and take a certain number of samples s for each algorithm, such that the system utility for each algorithm is reported for epochs $e \in \mathbb{N}, e \in [p, p+s)$. The values for p and s are selected by allowing MACRO-D sufficient time to stabilise, p, and taking that many samples after the first time that utility was found by MACRO-D. This gives us an indication of how each algorithm performs in the timescales that are sufficient for MACRO-D to find a high quality attractor. Note that the temporal position of the samples is only important for HN-EI and MACRO-D, since the behaviour of HN-C over multiple epochs is ergodic its mean response will not change over time. We measure all three datasets in the same manner nonetheless.

In the experiments here, we choose k=8, Z=15, N=Zk=120, τ=$\lceil 0.75T \rceil$=1172. These values are selected following a preliminary study on the modular case, which established the presence of multiple optima in each module with probability 0.956 $\pm$0.0127 of random instances for 8-bit modules, at the 95% confidence level. (In order to show any significance, there must be multiple optima otherwise the non-adaptive local search mechanism will be able to find the solution in each module trivially, and hence the overall solution – see for instance (Forrest and Mitchell, 1993b)). We use these parameters for a variety of r values across the full range. Each instance has random construction, so to measure general performance we run each algorithm on 100 different instances.

Hebbian Learning Can Resolve Unstructured Problems

MACRO-D and HN-EI discover higher utility states than the non-adaptive control, HN-C, throughout the range of rewiring values. For all values of r, both the mean and the maximum utility found by MACRO-D is higher than for HN-EI. The effect is strongest

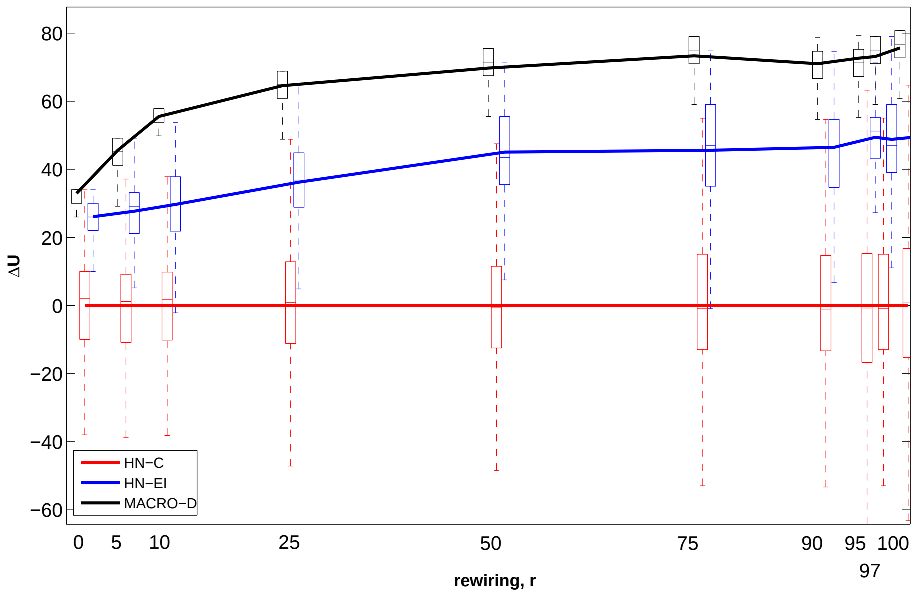


Figure 7.4: The distributions of difference in utility found by MACRO-D and HN-EI, relative to the mean utility level found by HN-C. Boxes indicate the inter-quartile range (the median line is shown in this box), whiskers show the minimum and maximum values in each distribution, and the solid line shows the mean values (connected to indicate the overall trend). Note that the data sets have different x-offsets to prevent any one set obscuring another.

for completely random sparse matrices and is sustained as modular organisation is introduced. It tails off when modules are completely separable (at $r = 0$), but is still statistically significant. We use the Wilcoxon rank sum test (Larsen and Marx, 2006) with confidence level α=0.001, and find that each of the distributions is statistically significantly different from one another at all values of r.

The results on the RRB problems demonstrate the that the improvement in quality of attractor discovered using the MACRO framework does not depend on the problem exhibiting systematic structure. Specifically, evolving covariance provides a significant improvement in optimisation even for problems with random construction and no systematic structure. Furthermore, this improvement is not restricted to systems within which all of the dependencies can be satisfied concurrently (*i.e.*, inconsistent problems).

7.5 Discussion

Together, the results from the above three experiments afford us an understanding of the abilities of a distributed implementation of the MACRO framework. In particular, we see that applying Hebbian learning to gradually update associations can provide efficient problem decomposition without the need for a centralised or batch-mode mechanism.

We also learn about the types of problem properties that MACRO-D can exploit, and using HN-EI as a substantive comparison highlights how unique these properties are. We discuss properties that distinguish these algorithms in more detail below. Finally, we also demonstrate that the MACRO concept can in principle provide good optimisation on a more general class of problem structures than we had seen in previous chapters – random pairwise matrix-based problems (which although cannot represent all problems are nonetheless a broad class).

The shift in perspective to consider the behaviour as a dynamical system is conceptually useful for several reasons. First, the evolution of associations in MACRO-D drastically changes the system dynamics. Not only is a global optimum found in nearly decomposable systems, but even after perturbing the system, simply following the (modified) dynamics reaches a global optimum again. That is, the modifications to the system increase the robustness of finding high utility states. Second, the prediction ability that comes from the evolved correlations in state changes increases the size of the basin of attraction of the global optima before it has been visited for the first time. Finally, even in a small system where optima can be enumerated, the modification to the dynamics can increase the likelihood of the global optima being visited. In such a case, we would not see a significant result in optimisation terms, and so this could be ignored. In a system that has too many optima to enumerate, the changes to dynamics can have more significant effects, both in terms of the robustness of high-utility attractor discovery, and in terms of optimisation quality.

A corollary of this chapter is that we have shown that the fundamental concepts that underpin the MACRO framework can be feasibly implemented in both centralised and decentralised forms. This is valuable because it helps us verify which features are important for success. Naturally, some features are merely design decisions where some option has to be taken, but which of the options is not important. We find the following conditions to be important for efficient optimisation by a MACRO approach:

1. Exploitation of the current units occurs on a fast timescale (micro-scale search);
2. The adaptation of those units occurs on a slow timescale relative to 1.;
3. Subsequent search in 1. is at the macro-scale. The process that arises from 1. and 2. can enable the prediction of unvisited, high utility states, provided that these are combinations of components that have made up the visited states.

We have investigated how both HN-EI and MACRO-D are capable of modifying the basins of attraction of a structured system to the extent that the global optimum is the only attractor visited. MACRO-D is a faster optimisation process than HN-EI, but also is important is how the experiments demonstrate the different methods of interpreting an evolved association. If a mechanism can increase the quality of an interaction when it happens, that is useful in some cases. Specifically, it can amplify common components from the local attractors that it visits to provide a form of modular exploration.[4] However, if we allow a change in association to enable correlated, multi-locus variation, we achieve a qualitatively distinct behaviour. Where the success of HN-EI depends on the strength of the between-module gradient with respect to the within-module gradient, the within-module gradient is only used in early relaxations in MACRO-D. Once modules have been appropriately formed, the strength of the between-module gradient just needs to be nonzero for a global optima to be easily discovered, as we saw for small weight ratios in experiment 1(a).

To illustrate the distinction between HN-EI and MACRO-D more concisely, let us compare the respective dynamics on the VSM problem (Mills and Watson, 2007b, introduced in Chapter 3) and the SBB problem (Watson and Jansen, 2007, first used in Chapter 5). Both problems have modular construction, wherein each module there are two locally optimal solutions. This gives rise to an exponential number of local optima in the system size, whose separation also scales with the system size. However, in the SBB problem, modules are separable, whereas in the VSM, modules are nearly-decomposable. MACRO-D can solve either type of problem, whereas HN-EI can only solve the VSM type of problem.

The initial system dynamics lead to significant correlations among the variable configurations within each module. Therefore, both algorithms reinforce the associations between variables in those sub-configurations. In the case of the VSM, there are also some correlations among the variable configurations between modules – and thus the basin of attraction for each global optima is slightly larger than the basin for each local optima. The between-module correlations are weaker than the within-module correlations, and so for HN-EI to reinforce the between-module weights to a level sufficient to change the dynamics to only recall the global optimum, it must visit enough patterns to reveal this signal accurately.

In the SBB problem, the absence of between-module dependencies means that there are no such between-module correlations in the local optima. Therefore, HN-EI will increase between-module weights as frequently as it decreases them, and there is no systematic route to change the size of the basin of attraction for the global optimum. Of course, a finite sample is unlikely to be perfectly representative and hence would have some (spurious) correlation that could be reinforced. But reinforcing coordinated modules would be just as likely as reinforcing anti-coordinated modules, and thus cannot be a systematic method for solving such a problem.

Now let us consider how the system dynamics change under the MACRO-D protocol. In both the VSM and the SBB, the strong within-module correlations lead to strong

[4]For further discussion of HN-EI, including the types of complex system in which our results are relevant, see (Watson et al., 2009b,a).

associations among the within-module variables. The effect of interpreting the evolved associations to inform multi-locus variation is to change the dynamics similarly for both problems. Specifically, because module solutions are varied all together, and accepted or rejected according to their collective improvement to the current context, MACRO-D searches the space of module solutions. In the VSM problem, the weak between-module dependencies provide a selective gradient to accept module solutions that increase global agreement. In the SBB problem, the highest utility solution to each module is independent of context. The dynamics of MACRO-D recover a global optimum from any initial condition in either problem. It does not rely on the presence of inter-module dependencies to cause the module-level components of the global optima more likely to co-occur in the local optima, in contrast to HN-EI.

The VSM is a natural problem to use for the Hopfield network models since it expressed as a sum of pairwise dependencies, and thus can use a single set of weights – at least for HN-EI. However, as discussed, there is a trade-off between instance difficulty and the quality of local optima relative to global optima in the VSM. In particular, the instances for which it is most difficult to find a global optimum offer only a small improvement in utility from a local optimum. The SBB does not have this same issue. Instead, it introduces higher order fitness bonuses for the module solutions, meaning that there is significant merit in solving a hard instance of this problem.

The contrast helps us to identify the type of problem where MACRO-D is a more effective method for problem solving. The distinction can be summarised as follows: HN-EI can recover a global optimum in systems where its basin is the largest of all the optima.[5] On the other hand, strong correlation in co-occurrence, even with low univariate occurrence, is sufficient to reinforce the likelihood of module formation in MACRO-D. Therefore, the module solution that is found most frequently after association adaptation need not have had the largest basin of attraction in the original dynamics, it only need be the highest utility solution.

The VSM is a suitable test system that can discriminate between the different modes of association utilisation (*i.e.*, MACRO-D from HN-EI) by varying the weight ratios. It is presently the case that none of the experiments in this chapter directly provide a tuneable distinction between MACRO-D and the non-associative control HN-C. However, it would be straightforward to confirm a polynomial versus exponential distinction in optimisation ability with a study on scalability (with respect to system size). Section 6.3.3 provides a result in this form for the batch-mode MACRO instantiations, whose successful operation is due to the same principles. We take an alternative approach to problem construction in the RRB. Here in the most modular case where the constraints are consolidated, evolution becomes easier for the simplest type of mechanism – because optimising each

[5]Note that although in problems like the VSM the global optima must have the largest basin of attraction, this basin is not necessarily large – it is just larger than the others (Fontanari, 1990; Watson et al., 2009a,b).

subfunction can be done in isolation. Conversely, with the VSM, introducing modular organisation through consolidating strong constraints leads to an increase in ruggedness in the landscape. These two problem constructions illustrate our discussion in Section 3.5 that the effect of introducing modular organisation depends on the original system.

The claims that we support in this chapter are threefold:

- Operationally, MACRO-D performs for reasons that are qualitatively equivalent to the explanation for MACRO-S:
  - Correlations in stable states discovered by micro-scale variation inform co-variance to change the system decomposition – creating macro-scale variation – and the modified decomposition reflects the structure of the problem.
  - MACRO-D can modify the system decomposition that is used by the rapid dynamics such that the highest utility attractor is found from any initial condition, when in nearly-decomposable environments (specifically demonstrated with the VSM);
- The substrates in which MACRO can provide adaptive multi-scale search include those where centralised control is not feasible;
- We demonstrate the optimisation ability to be more general than only nearly-decomposable problems. The new results are on problems with random structure and the capacity for frustrated dependencies.

These claims further support our thesis that adaptive multi-scale search is superior to any single scale of search in structured landscapes; and enrich the generality of this principle by demonstrating that it can be implemented in a distributed system on a broader class of test problems.

Chapter 8

Discussion, Implications and Conclusions

In this thesis, we have developed a technique that can automatically identify and exploit structure in a problem, and identified how the features and mechanisms of this technique make it an effective approach for optimisation. This chapter recaps our achievements and positions the thesis in a wider context. We first summarise the thesis, and follow this by evaluating aspects of the research that, if improved, could further strengthen our claims. Second, we make connections with, and discuss possible implications for, optimisation (Section 8.3) and evolutionary biology (Section 8.4). In Section 8.5, we identify the contributions made by this thesis. Section 8.6 concludes.

8.1 Summary of the Thesis

Our aim in this thesis was to develop a bottom-up approach to problem decomposition that is capable of efficient problem solving with no *a priori* knowledge of the problem structure.

We took an approach inspired by evolution, and borrowed concepts relating to symbiosis and symbiogenesis, and ecosystem functionality. Many existing evolutionary computation methods derive from the basic ideas of within-population processes that provide many variant solutions to a whole problem. We consider collective problem solving: organisms of different types perform heterogeneous tasks in an ecosystem. We draw upon this concept in the design of our algorithms, representing partial genotypes that ultimately embody candidate solutions for different parts of an overall problem.

Existing work on cooperative coevolutionary algorithms has laid important groundwork for collective problem solving, demonstrating that different sub-populations can cooperate to solve distinct sub-problems (Potter and De Jong, 1994). However, this work does not address the identification of appropriate decompositions except in a few limited or specialised cases. Estimation-of-distribution algorithms, in contrast, use sophisticated machine learning techniques that are capable of modelling epistatic dependencies in a problem, with no prior information regarding the problem structure (Pelikan et al., 2002). Our approach takes a different angle, with similarities to cooperative coevolutionary algorithms and compositional search algorithms, as well as model building from estimation-of-distribution algorithms.

Our two research goals were to develop an algorithm that provides automatic problem decomposition, and to identify the critical components of the proposed algorithmic approaches. To address our first research goal, we introduced algorithms that provide efficient optimisation through automatic problem decomposition. They automatically adapt the decomposition throughout a run, which affords search at higher scales. When there are several potential solutions to a sub-problem, each is retained and made available to higher-scale search. The higher-scale search identifies combinations of partial solutions that are compatible with one another, thus producing solutions of high overall quality. We used idealised cases that have simple modular structure to demonstrate that our method scales log-linearly, despite the fact that single-scale search methods require time exponential in the problem size.

To address our second research goal, we selected problems that have properties that are able to test a specific capability of our algorithms. Our experiments consequently enabled us to develop our understanding of the key characteristics, affordances and limitations of our proposed algorithms. In particular, we identified the importance of using the micro-scale search to guide adaptation in the decomposition, which in turn provides macro-scale search units. We conclude that separating the timescales for the fine-scale search and the adaptation of the decomposition is crucial for this kind of scalable optimisation.

8.2 Evaluation and Future Directions

Through developing the MACRO framework, we have provided a bottom-up approach to automatic problem decomposition, and have demonstrated it to be efficient in several classes of idealised test problems. In this section we identify several factors that limit the generality of our claims, and discuss how these issues could be addressed in future work.

We have demonstrated that automatically identifying a problem decomposition is considerably simplified when the decomposition adaptation mechanism is provided with locally optimal and noise-free contexts (Chapter 5). We further demonstrated that accurate decomposition is still possible even when there is some ambiguity in local optima (Chapter 6). However, we have not performed a detailed investigation of the sensitivity of MACRO to the period allowed to the fine-scale search, or in particular, how close to an optimum the fine-scale search achieves. We have some intuitions about the limiting cases. Without any separation of timescales, the contexts are arbitrary and extracting an accurate signal from such contexts is expensive and potentially only possible in limited cases (as we saw in Section 4.2). On the other hand, a separation in timescales beyond what is sufficient on average to reach a local optimum will reduce efficiency. Moreover, it may not be necessary for the hill climbing process to fully reach a local optimum.

Hence, a fuller investigation of the trade-offs relating to the separation of timescales would be valuable. Such a study would enable us to better understand how significant an influence the balance has on overall efficiency, and under what general landscape properties this influence is most significant. Furthermore, this form of investigation could also be carried out regarding the separation of timescales of mutation and crossover used in the genetic algorithm tested in Section 4.1 (albeit under tight linkage conditions).

Thus far, we have demonstrated a qualitative equivalence in the problem-solving abilities of MACRO-S and MACRO-D. However, a complete understanding of the different problem classes for which each of these algorithms are best suited remains an open question. In particular, it is desirable to better understand where the equivalence holds for adapting the decomposition in batch-mode (MACRO-H and MACRO-S) and adapting the decomposition in a sequential mode (MACRO-D). This may also have implications for the design of a decentralised estimation-of-distribution algorithm.

In Chapter 4 we demonstrated that the combination of linkage-preserving crossover and mutation is sufficient to solve modular problems such as the VSM, when the modularity is as strong as possible. We further demonstrated that search at a single scale is unable to solve these problems. Finally, we provided a logical argument as to why a combination of mutation rates would not be able to solve this type of modular problem. It remains an open question where in the scale of structural modularity for VSM instances that a combination of mutation rates might excel over a single mutation rate. More generally, it would be useful to more formally characterise the change in behaviour of uncorrelated search mechanisms for intermediate values of structural modularity in the VSM.

All three MACRO instantiations that we have investigated use very simple fine-scale search mechanisms to demonstrate their capabilities. While it is very likely that the most suitable choice will be domain-dependent, an investigation into how to match search mechanisms to problem domains is warranted. Any substitute must be extensible to meaningfully incorporate associations to inform multi-locus changes, but otherwise there are few restrictions.

One logical possibility is to use MACRO within each of the demes, creating a 'meta-MACRO' algorithm. Each lower-level instance would use the MACRO process to return a single high fitness configuration (or perhaps a small set). The meta-level algorithm would identify correlations amongst the configurations from all of the lower-level instances to define macro-level units. Further lower-level MACRO instances would then search in the reduced space of these re-defined units. Another pragmatic approach could be to run different search mechanisms in different demes, in an aim to provide greater confidence that particular variables are highly compatible.

One further aspect of the hill-climbing algorithm that warrants further investigation is the procedure used to create the macro-scale variation according to the decomposition. In MACRO-H, this is unambiguous as the units are re-defined. However, for the probabilistic algorithms, we have only used one method to create a macro-scale perturbation – using the association strengths from one particular focal node. There are several possible variants, which include a) taking an average of the symbiosis values for the current members of the group in deciding whether to include a new member; and b) chaining these so the most recent member gets to specify one addition. Again, the most appropriate choice is likely to be heavily influenced by the specific problem domain.

We have so far demonstrated that the principles of MACRO can provide superior optimisation using only simple mechanisms to model the decomposition (system structure). Indeed, we favour simpler mechanisms particularly when interpreting the processes as a biological model. However, there is scope to use some of the more powerful statistical learning methods used by contemporary EDAs. There are some trade-offs between the methods employed by MACRO-S or MACRO-D – including the cost of a further search in the model space (which in some cases is an NP-hard problem); and potentially the loss of a fully distributable mechanism. Nevertheless, understanding the limitations of a simple model with stable contexts is important for a full characterisation of MACRO.

8.3 Implications for Optimisation Techniques and Applications

8.3.1 Automatic Linkage Detection: A Comparative Perspective with EDAs

The issue of learning problem structures is fundamental in estimation of distribution algorithms, as it is in our symbiotic approach. We explore a number of dimensions to compare the properties of three types of algorithm: MACRO; a typical EDA (we take hBOA (Pelikan et al., 2006a) as a state of the art example where one is necessary); and the DSMGA+ (Yu, 2007; Yu and Goldberg, 2006).

We compare our approach with the DSMGA+ as they share a number of features (in particular, it is closest to MACRO-H). These include: 1) both DSMGA+ and MACRO-H build pairwise models that explicitly reduce dimensionality; 2) both algorithms are capable of modular variation, on account of 1); 3) both algorithms are recursive. There are important differences, most notably the single timescale of search. However, due to the three features listed above, we find it appropriate to consider in this section. The DSMGA+ is described further in Section 2.3.3.

Modular variation is a distinct possibility in EDAs, DSMGA+, and MACRO. Indeed, each approach aims to exploit structural information present (but unknown *a priori*) in the problem. We identify the mechanism for the generation of modular variation, and the subject of this variation. In a typical EDA, variation is generated by random sampling of the probabilistic model. Assuming some linkage has been modelled, particular variable combinations will be set at once, during the creation of a new individual. In the DSMGA+, variation is introduced by mutation and 'building-block-wise' crossover. These two operators are applied directly to selected individuals from the population. The interesting operator is the BB-wise crossover, assuming there has been a non-trivial clustering in that generation. In our symbiotic algorithms, the probabilistic model defines any multi-locus variation that may occur, and this affects the neighbourhood of state changes that are available in the dynamics followed by each deme.

The success of EDAs is often attributed to the application of sophisticated statistical learning techniques to extract plausible dependency structures. In contrast, the model in MACRO is adapted under a far simpler mechanism, but because the epistasis estimation is performed in locally optimal configurations, patterns are more coherent. When this form of model building is coupled with the fact that modelled dependency groups are employed in context, it need not be the case that a complicated statistical model represent the entire system dependencies. Instead, variation groups are accepted or rejected according to suitability, given the context.

We draw three contrasts between MACRO and EDAs and DSMGA+: 1) the frequency of model generation, or if there is a separation of timescales; 2) where variation is applied; and 3) how information regarding candidate genotypes is inherited.

As we have argued elsewhere (Section 8.3.2), the separation of timescales in adaptation of the decomposition, and utilisation of that decomposition is crucial for MACRO to provide efficient optimisation. EDAs do not use such a separation of timescales, instead generating a new model after a single round of selection amongst the fitter candidates drawn directly from the model. The DSMGA shares this property of EDAs.

An EDA samples all of the variables in its model to generate a new candidate at once. While weaker dependencies between variables in different modules in the model allow in principle many different combinations of module solutions, this would require an exponential sample size in the number of modules (that have multiple solutions). When associations have formed in MACRO, the within-deme dynamics operate at the macro-scale (*i.e.*, with multi-locus groups of variables), and selects for the most appropriate version of a module, given the current configuration. This allows candidates that are used in the associative update to be improved according to both the model and the context in that deme. Thus, the model need only be concerned with representing module solutions: context-sensitive search can determine (or improve) the compatibility of the combination of modules used. The use of an explicit and long-term population in the DSMGA makes the operation distinct from EDAs in this respect. Variation is applied after selection, by using 'building-block-wise crossover'. This allows the creation of a candidate that varies from its parent by the solution present for one module – as BB-wise crossover does not disrupt within-module settings (given an appropriate model). However, we contend that this type of variation is more disruptive because it will change half of the non-converged modules on average. Therefore an improvement resulting from the introduction of one module to a particular configuration may be masked by fitness losses through other module changes. This problem, coupled with the fact that BB-wise crossover is only applied once (for each candidate), leads us to conclude that neither EDAs nor DSMGA-like approaches apply the modular variation in as effective a manner as in our MACRO approach.

We highlight a final contrast regarding the inheritance of information in candidate genotype. One of the motivations in EDAs is to abstract away from an explicit population in order to avoid convergence (see Baluja, 1994). This is achieved by drawing an entirely new set of candidates from the modelled distribution in every generation, which massively reduces the correlation between one generation and the next when compared with a traditional GA. In MACRO, we share some of this motivation – maintaining diversity is key to finding all of the possible module solutions somewhere in the set of demes (or within a reasonable number of epochs in a serial instantiation). But we wish to temporarily retain all of the information possible regarding the context, in order to find the most harmonious combination of components. Therefore within an epoch, we use direct inheritance, which allows our algorithms to test the introduction of a single component at once; and between epochs, only inherit positional information through the epistatic model. The DSMGA sacrifices one of the advantages gained from using auxiliary dependency structure models by directly inheriting the population throughout a run. It is therefore likely to have difficulty with diversity maintenance, similar to tradition GAs (Mahfoud, 1992; Singh and Deb, 2006; De Jong, 1975). Indeed, the DSMGA+ uses an explicit diversity maintenance technique to mitigate such problems (Yu, 2007).

To summarise, we suggest that while the underlying aim of automatic problem decomposition is shared by our compositional approach and EDAs, the reasons for their success are distinct. In the case of EDAs, using powerful machine learning techniques to extract dependency patterns in large populations is sufficient to build accurate models. In the case of MACRO, employing multiple scales of search can provide scalable optimisation by emphasising the importance of selective contexts, thereby allowing the use of far simpler and more transparent dependency detection techniques.

8.3.2 Multiple Scales of Search

In this section, we discuss the significance of processes at multiple timescales, and the reciprocal influence between these processes for the performance of MACRO and for search processes more generally. We have attributed the success of several results in this thesis to the implementation of multiple scales of search. In MACRO, a fast, micro-scale search guides adaptation in the decomposition, which provides new macro-scale units for higher-order search (Chapters 5–7). Additionally, under the limited case of tight linkage assumptions, we demonstrated how a certain kind of GA can use the combination of a fast, small-scale search (mutation) and a slower, large-scale search (linkage-preserving crossover) to solve nearly-decomposable problems (Section 4.1).

Other methods also utilise multiple scales of search. For instance, models of the Baldwin effect (Hinton and Nowlan, 1987; Mills and Watson, 2005, 2006; Baldwin, 1896) complement genetic evolution with lifetime learning. The learning mechanism operates on a faster timescale than genetic evolution, and the two timescales are typically explicitly separated. Moreover, memetic algorithms apply local search to each candidate in their population (Hart, 1994; Krasnogor and Smith, 2005). Thus, several steps of small-scale search are applied between each application of crossover, which potentially introduces larger-scale changes.

Does search at multiple scales provide overall behaviour that is unavailable from the underlying components in isolation? In Chapters 5–7, we demonstrated that MACRO achieves qualitatively distinct results from search at a single scale. In certain cases, we have demonstrated a log-linear versus exponential advantage of MACRO when compared to the fast search scale alone (see Section 5.4). We further argued why applying only an association formation mechanism alone cannot solve problems efficiently, as it requires guessing both where the modules are and their configuration – a strategy that also scales exponentially with the problem size. A conceptual point made by models of the Baldwin effect (Hinton and Nowlan, 1987; Mills and Watson, 2006) is to explicitly demonstrate the necessity of multiple search scales. With only genetic adaptation, the fitness landscape makes discovering a higher fitness genotype an extremely rare occurrence. With only behavioural adaptation, fit phenotypes may or may not be discovered, which depends on the phenotypic plasticity and the genotype of an individual – both constant if no genetic evolution is allowed. Only when behavioural adaptation appropriately guides genetic evolution can a population move across fitness valleys (Mills and Watson, 2006).

We note a coupling between the fast and slow processes is central to the overall system behaviour in both MACRO and the Baldwin effect examples. The fast process introduces correlations to the configurations that the slow process acts upon; and changes in the slow process modify some aspect of how the fast process operates. Exactly what the slow process modifies differs in the Baldwin effect models and MACRO. Specifically, adaptations to the decomposition in MACRO change the degree of correlation between variables, such that particular allele combinations co-vary in higher frequency in later search. In the Baldwin effect models mentioned, genetic changes modify the (univariate) probability that a particular allele is present during subsequent lifetime learning. We can draw a contrast with coarse-graining in descriptions (see *e.g.*, Soetaert and Herman, 2009; Klotz et al., 1998). In these scenarios, it may be appropriate to model some processes by their equilibria, but there is only a one-directional influence. Thus, it is unlikely to enrich the overall result (except in simplifying model calculations).

There are almost certainly other scenarios that exhibit processes at multiple timescales, and algorithms that make use of multiple scales of search, either explicitly or implicitly. Whether the importance of principles that we have outlined above hold for other scenarios remains to be seen.

8.3.3 Extensions and Applications

The focus of this thesis has been to develop a principled understanding of how to automatically perform problem decomposition in a bottom-up manner. A natural next step is to apply this knowledge to practical computational problems. We have demonstrated our approach to be most effective when problems are nearly-decomposable. Thus, any application should begin with some examination of the domain to determine if there is likely to be significant localisation in interactions that a decompositional approach can gain some traction from.

In Chapter 7, we demonstrated that the principles can be feasibly implemented in a distributed system, and that it can efficiently solve static tasks. Accordingly, a second direction of interest is to explore how MACRO performs in the face of time-varying tasks. In particular, it is not entirely clear how MACRO would respond if the task was modified after the algorithm had narrowed its response to finding only global optima from the initial task. Relatedly, how do the principles of the MACRO framework apply for systems that undergo changes to their topology? Such an investigation would require a shift away from black-box optimisation assumptions which demand fixed length genotypes, but we believe that MACRO should still be applicable in principle.

We used several different idealised test problems to facilitate our investigations into the automatic discovery and exploitation of modularity. In Chapter 3, we introduced a flexible method to represent modularity. In Section 3.3 we identified three candidate dimensions that may provide a tuneable degree of (modular) difficulty:

1. Heterogeneity in the strength of interdependencies
2. Heterogeneity in the positioning of the classes of interdependency weights
3. Heterogeneity of the function that satisfies the interdependencies

We investigated the second of these dimensions in Chapter 3, and used the resulting landscape to test algorithms in the later chapters. The third dimension is also likely to present significant and systematic difficulty for optimisation, and merit further investigation. There is much still to do in elucidating the system properties under this type of modification: it would be of value to better understand the influence of such structuring on problem difficulty. Further investigation could answer questions such as: are there any methods that can still resolve the difficulty once a mixture of constraints are introduced? Are there particular ratios where the system becomes intractable for all methods?

8.4 Implications for Complex Biological Systems

While the main focus of this thesis has been in developing efficient algorithms, we are also interested in understanding the impact that associative evolution has on biological organisation. In this section, we discuss aspects of how the principles that underlie the MACRO framework may have biological relevance; highlight questions that modelling based on MACRO concepts raises; and identify connections with research in evolutionary sciences.

8.4.1 What does the MACRO framework suggest about associative evolution?

One way in which MACRO could be interpreted as a model of an evolving ecosystem is as follows. The ecosystem constitution adapts rapidly according to local *ecological* dynamics, symbiotic associations adapt gradually between co-occurring species in the ecosystem (*evolutionary* dynamics), and the associations in turn modify local dynamics. In the scope of this model, the kind of research questions that we can ask include: a) what kind of association formation mechanism can lead to the evolution of complexes that are unevolvable in the absence of associations? b) Under what conditions is such a distinction available?

We developed MACRO as an algorithm for problem-solving, and so it should be no surprise that some aspects, important for optimisation, are not fully appropriate in a biological context. The work in Chapter 7 shows that the principles of MACRO are distributable, such that associations are modified using only localised information. This algorithm (MACRO-D) forms the basis for the most plausible interpretation. Here, we identify and briefly discuss four assumptions that are relevant in this interpretation.

1. **An association increases the likelihood of co-dispersal such that multiple species may 'migrate' together.** MACRO-D clearly assumes that between-species interactions are evolvable: this is the focus of its operation. We refine this

assumption to demand a more specific type of evolvable interaction: to enable multi-locus variation in the local dynamics. Sterelny (2004) describes some examples that fit well with this assumption, for instance: “A leafcutter queen takes a sample of her fungus on her way to found a new colony”, (p13).

2. **Groups of multiple migrating species are selected on as a unit.** Migrant groups of multiple species are introduced simultaneously, and retained or rejected as a whole. We acknowledge that this may be unrealistic, particularly when symbioses are far from obligate. We discuss the relationship between variation and selective units further in Section 8.4.2.

3. **Associations change in a Hebbian direction.** In the experiments in Chapter 7, we simply assert the changes to association strengths, and investigate the optimisation ability when doing so. However, (Watson et al., 2009b) show mathematically that imposing Hebbian changes on associations gives the same result as selection on random variants. Specifically, a random variant whose association is in the Hebbian direction from the wild-type is selected for. Moreover, individual-based simulation models of (Watson et al., 2009c) and (Lewis, 2009) provide supporting evidence for the selection of Hebbian changes.

4. **Association strengths are symmetric.** In order that the network of MACRO-D has only fixed point attractors, we initialise the systems to have symmetric association strengths (*i.e.*, $S_{ij} = S_{ji}$). This assumption is unrealistic, and may be too strong even to provide the qualitative results obtained so far. We are yet to investigate other forms of interaction and hence other categories of system attractor.

If we take all of the above assumptions to be feasible, we can now interpret the results obtained from simulated experiments in Section 7.4 to inform us about some scenarios under which macro-evolution can be qualitatively distinct from micro-evolution. Three conditions that are present in our results are:

1. **Ecosystems have many attractors.** In a unimodal environment, where the intrinsic ecological dynamics already lead to configurations where many dependencies are satisfied. Thus, it is not straightforward to see how the evolution of associations might change this situation.

   Empirical studies suggest that there exist multiple stable states in natural biosystems (Thompson, 1988; Sutherland, 1974), as well as theoretical arguments for the presence of multiple peaks in the fitness landscape of individual populations, created by significant epistasis (Whitlock et al., 1995).

2. **Ecosystems receive frequent and significant perturbations, such that they visit many attractors.** For the evolved associations to provide modular

variation, it is important that the ecosystem visits multiple attractors. If only a single attractor is visited, under our model, the likelihood of this attractor state will be reinforced. However, as for (1), the resultant ecosystem state is unlikely to be distinct in the absence or presence of evolved associations.

3. **The frequency of perturbations is low when compared to the ecological dynamics.** It is important that the time spent at attractors dominates the length of the transients in ecological dynamics, such that associations are selected for in non-arbitrary conditions.

While conditions (2) and (3) are somewhat in tension, we have not carefully tuned the period between perturbations in our experiments (see Section 8.2). We highlight these conditions here to flag an area for future investigation. Note however, that the body of work described by Hogeweg (2007) considers the timescales for ecological and evolutionary processes to be separated, but not so widely separated to rule out important mutual interactions between each process (see in particular Laan and Hogeweg, 1995).

The results in Section 7.4 suggest that in an abstract landscape whose interdependencies exhibit near-decomposability, associative evolution can lead to rare and adaptively significant complexes that are unavailable via non-associative evolution. Naturally, a more thorough treatment is necessary to better understand the algorithmic leverage provided by associative evolution.

8.4.2 The Relationship Between Units of Variation and Units of Selection

In our algorithms, macro-scale variation groups are created probabilistically, with correlations biased according to the current decomposition. A group is selected on as a whole: if the overall system utility is improved, the group remains in the ecosystem, and if not, the entire group is rejected. Conceptually, this effectively causes the units of variation to be synonymous with the units of selection.

An alternative scheme might allow groups to migrate together, but select on the suitability of individual variable allocations given the new context. We suggest that because the individual selection is performed in the new context, with the entire migration group, the ultimate changes in ecosystem composition will not be significantly different than if selecting on groups as an entire unit. Recall that migration groups typically comprise species that were frequently found to co-occur in locally stable contexts. Thus, the individual species would be selected in the context of the particular group.

We would like to test the hypothesis that both of these modes of selection have qualitatively similar results. We note that this distinction is closely related to concepts of the existence of individuality without a formal boundary (Penn, 2002), as well as the relationship between symbiosis and symbiogenesis.

8.4.3 Mechanisms for Increasing the Stability of Selective Contexts

The principles that we explore with the MACRO framework revolve around reinforcing the co-occurrence of entities, such that when entities are found to co-occur frequently their chance of future co-dispersal is increased. This reinforcement (adaptation of associations) leads to the evolution of fit associations when selected upon in stable (locally optimal) contexts (Mills and Watson, 2008, 2009).

We can view a symbiotic association as improving access to a particular resource, which stabilises an organism's selective context. This is most clearly illustrated in symbiogenesis: a symbiont is encapsulated such that both genomes are inherited with high fidelity. That is, some aspects of the environments of the previously free-living entities are highly stabilised. Niche construction (Odling-Smee, 1995) typically describes organismic traits that modify abiotic factors (such as nests; soil properties). In contrast, we recognise symbiosis as a mechanism that modifies biotic factors (Powers et al., 2009; Penn and Harvey, 2004).

The results of controlling the stability of ones environment are broad – for instance, the internalised thermo-regulation that mammals and birds possess (contrast this with the environmental dependence that reptiles and amphibians have with regards to temperature, Lavers, 2000). This is a type of homeostasis, (Williams, 2006), which may give rise to a different type of stability than symbioses do.

We wish to further explore the connections between symbiosis as a mechanism for increasing biotic stability and mechanisms that provide homeostatic control of biotic and abiotic factors, as well as the influence of both categories of stabilising mechanisms in evolution.

8.4.4 Multi-scale Processes in Adaptive Social Networks

In Chapter 7 we implemented MACRO-D using an abstract complex network in which both the pattern of interactions, and the states of the nodes adapt. Viewing MACRO in this way highlights similarities to models from the domain of adaptive networks (Bornholdt and Rohlf, 2000; Gross and Blasius, 2008). This work concerns dynamical processes that occur on networks where the networks themselves also adapt, and in particular, the mutual interactions between adaptation of the topology and adaptation of the network states. Santos et al. (2006); Van Segbroeck et al. (2009) consider how cooperation can evolve when the agents in the social network can directly modify with whom they interact. Geard and Bullock (2008) investigate how social network topology influences affiliation group formation, and vice versa. Interestingly, this model periodically seeds affiliation groups that are allowed to equilibrate before any network connections are modified. In MACRO, it is crucial that the state dynamics are periodically perturbed, such that interactions are adapted to generalise over many different attractors.

A key feature of the algorithms presented in this thesis is the separation between the timescales on which search is performed with the current units and modification of those units for future search. Similarly, timescale separation is an important aspect of adaptive networks: when the network topology is changed slowly, but in response to properties of the (faster) dynamics on that network, the overall system response can self-organise to a critical state (Bornholdt and Rohlf, 2000). Understanding the effects of timescale separation on system behaviour is an open challenge in the field of adaptive networks. Further exploration of the relationship between these models and the concepts in MACRO is likely to prove fruitful.

8.5 Summary of Contributions

8.5.1 Supporting Contributions

In Chapters 3 and 4 we make contributions that lay down groundwork of this thesis, thereby supporting the development of the family of algorithms that comprise our primary contributions. We outline three specific contributions below.

A natural and flexible problem definition that exhibits modular interdependency. In Chapter 3, we introduced a generator whose problem instances exhibit a flexible amount of modular interdependency. Our investigation of the resultant landscape helps us to understand that ruggedness need not equate to irreducible complexity. Specifically, this landscape features structured epistasis that is in principle resolvable via a search mechanism that can provide variation that corresponds to the modular structure.

A demonstration that multiple search scales in a genetic algorithm can decompose a space, if tight linkage is assumed. Under the special assumption of knowledge of the variable ordering (‘tight linkage’), we demonstrated that certain genetic algorithms can in principle exploit multiple scales of structure in a problem that exhibits modular interdependency (see Section 4.1). When a fast scale of variation (mutation) finds module solutions, and a diversity maintenance mechanism preserves the different solutions for each module (deterministic crowding), then linkage-preserving crossover can explore combinations of modules to find optimal top-level solutions.

A demonstration of how reciprocal synergy measures can provide problem decomposition. In Section 4.2 we described an approach to problem decomposition that is not dependent on any prior knowledge of the epistatic linkage structure. Instead, the algorithm discovers this structural information on the fly. This approach uses a simple representation that allows us to interrogate the decomposition structure that has been discovered. It reveals that the algorithm can identify the structure of modular problems, and uses this information to scale up the units of search in a compositional manner.

8.5.2 Primary Contributions

The central contribution made by this thesis is the development of a family of algorithms that are capable of automatic problem decomposition for scalable search. We identify the contributions made through the development of each of the three variants that comprise this family of algorithms.

MACRO: A scalable optimisation method for nearly-decomposable problems

We introduced the MACRO framework, which is founded on the following principles to provide problem decomposition: a) rapid micro-scale search guides a slower adaptation of the decomposition; b) changes in the decomposition scale up the search to explore *combinations* of module solutions; c) the entire process is applied recursively such that an improved decomposition enables the discovery of further structure, and so on; and d) the mechanism that adapts the decomposition reinforces strong co-occurrence of lower-level units at local optima, creating higher-level units that encapsulate (or reflect) module solutions.

We developed three implementations of this framework in Chapters 5, 6 and 7. Each implementation improves our understanding of where the principles used by MACRO are applicable, as well as the mechanisms that are able to facilitate the approach.

An elucidation of the problem properties that MACRO can exploit

We demonstrated MACRO to be capable of solving several classes of problem, each selected to test different facets of its abilities. We use the features that are most relevant for problem decomposition to categorise the problems into three classes: 1) nearly decomposable, neat modular structures; 2) ambiguity in local optima, with multiple legitimate decompositions; and 3) inconsistent problems with no systematic structure.

Neat and idealised modular structure. The first class of problems that we investigated have neatly defined modular structures. These problems exhibit very rugged landscapes with only very few global optima in an exponential number of local optima, whose local optima are far apart. These features present pathological difficulty for single scale search methods that cannot decompose the problem: they perform at least as badly as $\Omega\left(2^{\sqrt{N}}\right)$, where N is the problem size. However, in Chapter 5 we demonstrated that MACRO is capable of adapting the decomposition to reflect the modular structures, thereby solving the problems very efficiently. The neat formulation of these problems allow us to formally characterise the behaviour of MACRO, and we demonstrated that MACRO-H scales as $\Theta\left(N \log N\right)$ on problems with both single-scale (SBB) and hierarchically structured (HIFF) modularity. Moreover, on the hierarchical problem, MACRO-H outperforms hBOA (a state of the art Bayesian network model-building algorithm that scales as $\mathcal{O}\left(N^{1.55}\log N\right)$, Pelikan, 2002), despite the simplicity of our method.

Problems that exhibit ambiguity in the local optima. The second class of problems that we investigated moved away from the idealised situation above, to exhibit ambiguity in the information available from local optima. In Chapter 6 we first used a directed test problem to highlight this issue and to reveal why it is a difficulty for our initial method (MACRO-H). Moreover, we used an Ising model based problem (RBIM) which accentuates the difficulty. Unlike the neatly defined modular problems, the RBIM has overlapping module boundaries. Consequently, the signal from correlations across several local optima are blurred out and automatically identifying a suitable decomposition is made more difficult.

Nevertheless, we developed MACRO-S, which is able to use the noisier information in the local optima to probabilistically define correlated macro-scale units for future search. We demonstrated that the use of such probabilistic associations is capable of efficiently solving problems with ambiguity in the local optima, even though the abundance of local optima cause significant difficulty for local search methods.

Problems with conflicting sub-solutions, and no systematic structure. The third type of difficulty that we investigated involved problems with inconsistent solutions, and no systematic structure. The two problem categories described above are consistent: there exists a configuration where all of the dependencies are resolved (*i.e.*, global optima). The RRB problems that we considered in Chapter 7 offer no guarantee of consistency. Instead, sub-solutions can conflict with all possible sub-solutions in other components. Moreover, for the fully randomised parameterisation of the RRB problem, the instances exhibit no systematic structuring in the interdependencies.

These features contrast with the idealised modular structures where we might, in principle, expect a decompositional approach to be able to offer significant gains. Without any systematic structure, it is not obvious that automatic problem decomposition should be able to confer a benefit. However, in Chapter 7 we demonstrated that probabilistic macro-scale search provides significant improvements in optimisation even under these conditions.

In sum, we have demonstrated that MACRO is versatile with respect to the types of problem characteristic that it can optimise.

An understanding of the critical features in MACRO

We developed three algorithms under the MACRO framework that are progressively simpler and more general. Each offers a different piece of the picture in terms of understanding the affordances and limitations of the approach. Chapter 5 introduced MACRO-H, and illustrated where the concepts offer the strongest algorithmic leverage. Chapters 6 and 7 introduced two probabilistic variants of the algorithm, MACRO-S and MACRO-D. We took forward the principles of identifying and encapsulating lower-level solutions for use as macro-scale units in higher-level search. Where MACRO-H uses explicit encapsulation, we demonstrated that probabilistically introducing correlations in future variation can provide abstraction of lower-level components in a more general manner. Both MACRO-H and MACRO-S explicitly analyse a batch of local optima to adapt the decomposition and provide macro-scale units. With MACRO-D, we demonstrated that this too is unnecessary to provide macro-scale search. Instead, gradually reinforcing co-occurrence at each local optimum visited in sequence can provide qualitatively equivalent results. We showed that a mechanism as simple as Hebbian learning can facilitate the evolution of macro-scale units that correspond to the system structure.

By considering the investigation of these algorithms together, we arrive at a comprehensive understanding about the abilities of evolved macro-scale search for problem decomposition. Overall, the two most fundamental features that MACRO depends upon are:

- Forming structures that provide macro-scale units of variation through reinforcing correlations between components that appear *at or near local optima*; and
- Exploiting the current units on a more rapid timescale than forming new macro-scale units, thereby generalising over many local optima to create macro-scale units that reflect the problem structure.

These are general conditions that render the MACRO concepts applicable in distributed networks, within which all adaptation uses only localised information. The resulting process, though simple, nevertheless offers powerful optimisation through automatic problem decomposition.

8.6 Conclusions

A divide and conquer approach to problem solving offers much promise: splitting a problem into smaller sub-problems that are easier to solve, then assembling sub-solutions into a full system solution can be far more efficient than solving the problem as one monolithic whole. Indeed, this type of approach seems straightforward to apply when we know enough about a problem to specify a suitable decomposition – that is, to identify what the smaller sub-problems are. However, the required decompositional information is typically not available *a priori*. Thus, the challenge that we addressed in this thesis is to understand how decompositions that enable efficient search can be discovered and exploited.

The MACRO framework we present in this thesis can solve, in time sub-quadratic in the problem size, nearly decomposable problems that require exponential time for any local search method. Our method uses micro-scale search to guide the formation of structures that enable macro-scale search over several scales of organisation, effectively decomposing the problem in a bottom-up manner without *a priori* information about the problem structure.

Appendix A

Properties of the Variable Structural Modularity Problem

In this appendix we characterise some aspects of the VSM, as introduced in Chapter 3 (see also Mills and Watson, 2007b). In particular, we consider aspects that relate to local optima, and the conditions for which particular configurations are locally optimal. The three characteristics that we elaborate on are:

- The number of optima
- The positions of optima
- The weight ratio for which local optima are present

For the first two properties, we initially identify the positions of local optima within a module (A.1), and then extend to the general case with multiple modules (A.2). We derive the third property in Section A.3. In this chapter, we assume that the internal weights are tightly clustered such that the modularity is as strong as possible.

A.1 Number and Position of Local Optima In a Single Module

The fitness contribution of a single module, is given by the Hamiltonian:

$$H(\boldsymbol{\sigma}) = \sum_{\langle i,j \rangle} \sigma_i \sigma_j J_{ij}, \tag{A.1}$$

where $\boldsymbol{\sigma}$ is a state vector of N binary variables, $\sigma_i \in \{-1, +1\}$, and J_{ij} is the interaction strength between σ_i and σ_j. The system is fully connected, though the interaction

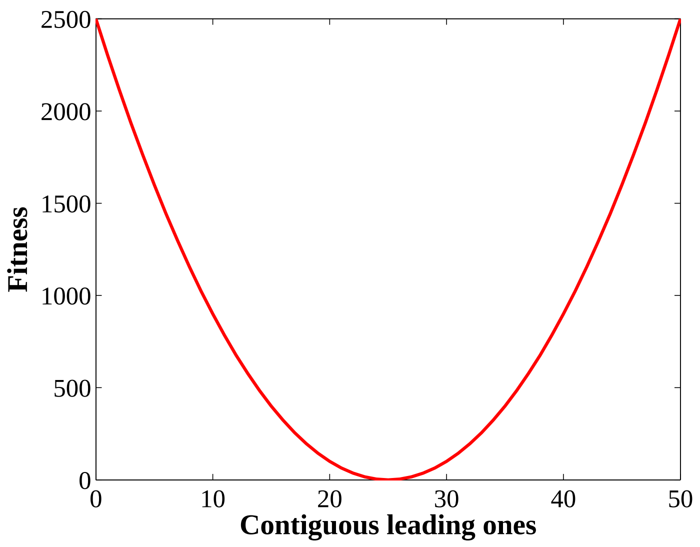


Figure A.1: A cross section through the fitness landscape for a single module in the VSM problem with k=50 and w_i=1.

strengths are not homogeneous. Within a module in the VSM, $J_{ij} = w_i$, and all other interactions have strength w_e. Here, we consider a single module system, and thus $\forall J_{ij} = w_i$. Since every site is connected to every other site in the module, this can be reformulated as a function of unitation (*i.e.*, the number of ones, or up-spins, in the module), without considering the configuration specific to any particular sites:

$$H(\boldsymbol{\sigma}) = k^2 + 4U(U-k), \tag{A.2}$$

where k is the number of variables in the module, and U is the unitation count. This simple form shows the fitness to have a quadratic relationship with unitation.

To find the minima we differentiate Equation A.2,

$$H' = 8U - 4k, \tag{A.3}$$

and by setting to zero, we find that the minima is at $U = k/2$. That is, at exactly spin-up and half spin-down.

To find the maxima, by inspection we see that the equation is of a quadratic form. Thus, in any interval there can be either one or two maxima (depending on if there is a turning point in the interval). We established that there is a turning point within the interval $0 \leq U \leq k$, and thus expect two maxima. By inspection of figure A.1 these are at all-down and at all-up configurations. This is straightforwardly reasoned about: when all of the variables agree, all of the dependencies are satisfied and this leads to a maximal fitness value.

More formally, we consider the limits of the function and its interval. The limits of unitation, U, are at $\lim_{min} = 0$ and $\lim_{max} = k$. For each of these limits, the value of H cannot exceed k^2. This occurs when the $4U(U-k)$ term of Equation A.2 is zero, which is possible either when $U = 0$ or when $U - k = 0$, namely when $U = k$.

A.2 Number and Position of Local Optima with Multiple Modules

The simplest way to intuit about these properties is to consider a case where w_e is set to zero, which leaves Z independent modules. As we saw in the previous section, there are 2 optima per module. When there are no interactions between modules, each of the modules can take on any of their optimal configurations. This gives rise to an exponential number of system-level optima (*i.e.*, there are 2^Z local optima, since $Z \propto N$). Since under these conditions both solutions to each module are equally fit, all local optima are also globally optimal.

However when external interactions exist there is a pressure towards modules agreeing, in addition to the pressure towards variables agreeing within the module. Therefore for any value of $w_e > 0$, the massive degeneracy is broken and exactly two solutions become globally optimal: the all-down and all-up configurations. (See the next section for a discussion of the conditions under which all 2^Z configurations are in fact locally optimal).

For the local optima that are not globally optimal, there are several configurations that lead to the same fitness level. This is due to the symmetry in the organisation of the between-module weights: for a given number of modules set to the all-up optima, and the remaining modules set to the all-down optima, the overall fitness is not affected by which of the modules is in which class. Hence, the number of different configurations that provide the same fitness value is binomially distributed. Note that there is an additional symmetry in the organisation: since the fitness contribution is the same if all states are flipped, we can simply consider the majority in the number of modules that are solved in the same direction. Thus, there are $Z/2 + 1$ different fitness levels that local optima can take in the VSM. The density of configurations at each fitness level increases with distance from the global optima. This means that the largest proportion of locally optimal configurations give the lowest fitness local optima.

A.3 Thresholds for Presence of Optima in the VSM

In the previous section, we identified the positions of local optima – the only configurations that could possibly be locally optimal. However, strong external dependencies can overpower some internal dependencies and cause some of these configurations to no longer be locally optimal. In this section, we derive the ratio at which each class of optima become locally optimal above, but are not below.

Let us assume that all of the modules are at one solution or the other (all-down or all-up). Further, let us assume that of Z modules, $Z-1$ are solved at all-up, and 1 is at all-down. To test if this position is locally optimal, we must consider the three configurations:

```
A. 111 111 ...  110 000
B. 111 111 ...  111 000
C. 111 111 ...  111 100
```

If internal weights are strong (*i.e.*, $w_i \gg w_e$), $f(B) > f(C)$, $f(B) > f(A)$. If internal weights are weak (*i.e.*, as $w_i \to w_e$), it is still the case that $f(B) > f(A)$. However, $f(B)$ may not exceed $f(C)$. Thus, we compare $f(B)$ and $f(C)$, in terms of Z and k.

Our approach is to identify the differential fitness contributions between configurations B and C, and the threshold is at the point where this differential is zero (so above the threshold, configuration B is locally optimal). As we shall see, the change in the last module results in the following modifications in fitness: 1) a loss in internal dependencies satisfied; and 2) An increase in external dependencies satisfied.

The fitness of configurations B, $f(B)$, and C, $f(C)$, are given by:

$$f(B) = Zk^2 w_i + (Z-1)(Z-2)k^2 w_e, \tag{A.4}$$

$$f(C) = \left[Zk^2 + 2(1-k)\right] w_i + \left[(Z-1)(Z-2)k^2 + 2k(Z-1)\right] w_e. \tag{A.5}$$

We find the differential fitness contribution:

$$f(B) - f(C) = 2(1-k) w_i + 2k(Z-1) w_e. \tag{A.6}$$

Setting to zero, and rearranging,

$$\begin{aligned} 2(1-k) w_i + 2k(Z-1) w_e &= 0, \\ k(Z-1) w_e &= (k-1) w_i. \end{aligned} \tag{A.7}$$

Rearranging to describe w_i in terms of k, Z, w_e, we arrive at the equality:

$$w_i = \frac{k(Z-1)}{k-1} \cdot w_e. \tag{A.8}$$

For ratios such that w_i is greater than this value, this configuration will be locally optimal.

Generalised Threshold for Optima

Now we extend this derivation to a more general case, where an arbitrary number of modules are solved to all-up, with the remainder solved to all-down. Let us again consider cases B and C. This time, there are X modules solved to all-up, and $Z - X = Y$ modules solved to all-down in configuration B. Configuration C has one bit changed towards all-up – again, a decrease in internal dependencies satisfied, and an increase in external dependencies satisfied. W.l.o.g. we constrain $X \geq Y$. That is, in the following derivation we assume that the changed locus is towards the majority of the entire configuration.

```
B. 111 111 ...  111 000 000 ...  000
C. 111 111 ...  111 100 000 ...  000
```

The fitness of configuration B, $f(B)$, in terms of w_i, w_e, X, Y, k and Z is given by:

$$f(B) = Zk^2 w_i + \left[X(X-1)k^2 + Y(Y-1)k^2\right] w_e. \tag{A.9}$$

The fitness of configuration C, $f(C)$, in terms of w_i, w_e, X, Y, k and Z is given by:

$$f(C) = \Big[(Z-1)k^2 + 1^2 + (k-1)^2\Big] w_i + \tag{A.10}$$
$$\Big[X(X-1)k^2 + 2Xk + (Y-1)(Y-2)k^2 + 2k(k-1)(Y-1)\Big] w_e. \tag{A.11}$$

We aim to identify the threshold for which $f(B) = f(C)$. We label the terms modified from $f(B)$ as LHS, and the terms modified from $f(C)$ as RHS from here on. We begin by subtracting $(Z-1)k^2 w_i$, and $X(X-1)k^2 w_e$ from both sides,

LHS:
$$= k^2 w_i + Y(Y-1)k^2 w_e. \tag{A.12}$$
RHS:
$$= \Big[1^2 + (k-1)^2\Big] w_i + \Big[2Xk + (Y-1)(Y-2)k^2 + 2k(k-1)(Y-1)\Big] w_e. \tag{A.13}$$

Expanding and collecting terms in RHS:

RHS:
$$= \Big[k^2 + 2 - 2k\Big] w_i + \Big[2Xk + Y\left(Yk^2 - 2k^2\right) - \left(Yk^2 - 2k^2\right) + 2Yk^2 - 2k^2 - 2Yk + 2k\Big] w_e$$
$$= \Big[k^2 + 2 - 2k\Big] w_i + \Big[2Xk + Y^2k^2 - Yk^2 + 2Yk + 2k\Big] w_e. \tag{A.14}$$

Subtracting all LHS terms from both sides of Equation A.12 and A.14, we obtain a term that describes the difference in fitness between B and C:

$$[2-2k]\,w_i + [2Xk + 2k - 2Yk]\,w_e = 0. \tag{A.15}$$

Rearranging to describe w_i in terms of k, X, Y, w_e, we arrive at the equality:

$$w_i = \frac{Xk + k - Yk}{k-1} w_e, \tag{A.16}$$

which identifies the threshold of the weight ratio at which a particular configuration becomes locally optimal. Equation A.16 agrees with Equation A.8 for $X = Z - 1, Y = 1$.

The first class of optima to disappear, with increasing external weights, is at $X = Z - 1$. The corresponding ratio is $\frac{w_i}{w_e} = \frac{(Z-1)k}{k-1}$. The last class of local optima to disappear are those maximally distant from the global optima, with half of the modules at all-down and half at all-up solutions. The corresponding ratio is $\frac{w_i}{w_e} = \frac{k}{k-1}$.

Bibliography

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985) A learning algorithm for boltzmann machines. *Cognitive Science*, **9**:147–169.

Arthur, W. B., Durlauf, S. N., and Lane, D. A. (eds.) (1997) *The economy as an evolving complex system II.* Addison-Wesley.

Bäck, T. (1996) *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford University Press.

Baldwin, C. Y. and Clark, K. B. (2000) *Design Rules, Vol 1: The Power of Modularity.* MIT Press, Cambridge, MA.

Baldwin, J. M. (1896) A new factor in evolution. *American Naturalist*, **30**:441–451.

Baluja, S. (1994) Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. *Tech. Rep. CMU-CS-94-163*, Carnegie Mellon University, Pittsburgh, PA.

——— (1997) Genetic algorithms and explicit search statistics. *Advances in Neural Information Processing Systems 9*, Mozer, M., Jordan, M., and Petsche, T., eds., pp. 319–325, MIT Press, Cambridge, MA.

Baluja, S. and Caruana, R. (1995) Removing the genetics from the standard genetic algorithm. *The International Conference on Machine Learning*, Prieditis, A. and Russel, S., eds., pp. 38–46, Morgan Kaufmann, San Mateo, CA.

Baluja, S. and Davies, S. (1997a) Combining multiple optimization runs with optimal dependency trees. *Tech. Rep. CMU-CS-97-157*, Carnegie Mellon University, Pittsburgh, PA.

——— (1997b) Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. *International Conference on Machine Learning*, pp. 30–38, Morgan Kaufmann, San Francisco, CA.

——— (1998) Fast probabilistic modeling for combinatorial optimization. *Fifteenth National Conference On Artificial Intelligence (AAAI-98)*, pp. 469–476, AAAI, Menlo Park, CA.

Beaumont, H. J. E., Gallie, J., Kost, C., Ferguson, G. C., and Rainey, P. B. (2009) Experimental evolution of bet hedging. *Nature*, **462**:90–93.

Beyer, H.-G. and Schwefel, H.-P. (2002) Evolution strategies - a comprehensive introduction. *Natural Computing*, **1**(1):3–52.

Blum, C. and Roli, A. (2003) Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Computing Surveys*, **35**(3):268–308.

Bogorad, L. and Deem, M. W. (1999) A hierarchical approach to molecular evolution. *PNAS*, **96**:2591–2595.

Bornholdt, S. and Rohlf, T. (2000) Topological evolution of dynamical networks: Global criticality from local dynamics. *Physical Review Letters*, **84**(26):6114–6117.

Borrett, S. R., Fath, B. D., and Patten, B. C. (2007) Functional integration of ecological networks through pathway proliferation. *Journal of Theoretical Biology*, **245**(1):98–111.

Boyan, J. A. (1998) *Learning evaluation functions for global optimization*. Ph.D. thesis, Carnegie Mellon University.

Boyan, J. A. and Moore, A. W. (2000) learning evaluation functions to improve optimization by local search. *Journal of Machine Learning Research*, **1**:77–112.

Bull, L. and Fogarty, T. C. (1995) Artificial symbiogenesis. *Artificial Life*, **2**(3):269–292.

Bull, L., Fogarty, T. C., and Pipe, A. (1995) Artificial endosymbiosis. *Procs ECAL*, Moran, F. and Mereno, A., eds., pp. 273–289, Springer.

Callebaut, W. (2005) The ubiquity of modularity. *Modularity*, Callebaut, W. and Rasskin-Gutman, D., eds., pp. 3–28, MIT Press.

Chen, Y.-p., Yu, T.-L., Sastry, K., and Goldberg, D. E. (2007) A survey of linkage learning techniques in genetic and evolutionary algorithms. *Tech. Rep. 2007014*, Illinois genetic algorithms laboratory, Urbana, IL.

Ciliberti, S., Martin, O. C., and Wagner, A. (2007) Robustness can evolve gradually in complex regulatory gene networks with varying topology. *PLoS Computational Biology*, **3**(2):e15.

Clune, J., Beckmann, B. E., Pennock, R. T., and Ofria, C. (2009) HybrID: A hybridization of indirect and direct encodings for evolutionary computation. *Proceedings of the European Conference on Artificial Life (ECAL)*, Kampis, G. and Szathmary, E., eds., Springer.

Clune, J., Ofria, C., and Pennock, R. T. (2008) How a generative encoding fares as problem-regularity decreases. *Proceedings of the 10th International Conference on Parallel Problem Solving From Nature*, pp. 358–367, Springer.

Coffin, D. and Smith, R. E. (2007a) The limitations of distribution sampling for linkage learning. *Proceedings of the IEEE Congress of Evolutionary Computation 2007*, pp. 364–369.

——— (2007b) Why is parity hard for estimation of distribution algorithms? *Procs GECCO*, Lipson, H. and Thierens, D., eds., p. 624, ACM Press.

Coyne, J. A., Barton, N. H., and Turelli, M. (1997) Perspective: A critique of sewall wright's shifting balance theory of evolution. *Evolution*, **51**:643–671.

Daniels, M. G., Farmer, J. D., Gillemot, L., Iori, G., and Smith, E. (2003) Quantitative model of price diffusion and market friction based on trading as a mechanistic random process. *Physical Review Letters*, **90**(10):108102.

Darwen, P. J. and Yao, X. (1997) Speciation as automatic categorical modularization. *IEEE Transactions on Evolutionary Computation*, **1**(2):101–108.

Dauscher, P., Polani, D., and Watson, R. A. (2006) A simple modularity measure for search spaces based on information theory. *Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems (ALifeX)*, Rocha, L. M., Yaeger, L. S., Bedau, M. A., Floreano, D., Goldstone, R. L., and Vespignani, A., eds., pp. 344–349, International Society for Artificial Life, MIT Press.

de Boer, F. and Hogeweg, P. (2007) The role of speciation in spatial coevolutionary function approximation. *Procs GECCO*, Lipson, H. and Thierens, D., eds., pp. 2437–2441, ACM Press.

De Bonet, J. S., Isbell, C. L., and Viola, P. (1997) Mimic: Finding optima by estimating probability densities. *Advances in Neural Information Processing Systems*, Mozer, M., Jordan, M. I., and Petsche, T., eds., vol. 9, pp. 424–431, MIT Press, Cambridge, MA.

de Jong, E. D. and Thierens, D. (2004) Exploiting modularity, hierarchy, and repetition in variable-length problems. *GECCO*, Deb, K. e. a., ed., pp. 1030–1041, Springer, Berlin.

de Jong, E. D., Thierens, D., and Watson, R. A. (2004) Defining modularity, hierarchy, and repetition. *GECCO Workshop on Modularity, regularity and hierarchy in open-ended evolutionary computation*, pp. 2–6.

de Jong, E. D., Watson, R. A., and Thierens, D. (2005a) A generator for hierarchical problems. *GECCO Workshop on the Theory of Representations.*

——— (2005b) On the complexity of hierarchical problem solving. *Genetic and Evolutionary Computation Conference*, pp. 1201–1208, ACM Press.

De Jong, K. A. (1975) *an analysis of the behaviour of a class of genetic adaptive systems.* Ph.D. thesis, University of Michigan, Department of Computer and Communication Sciences, Ann Arbor.

Defaweux, A. (2006) *Evolutionary transitions as a metaphor for compositional search.* Ph.D. thesis, Vrije Universiteit Brussel.

Defaweux, A., Lenaerts, T., van Hemert, J., and Parent, J. (2005) Transition models as an incremental approach for problem solving in evolutionary algorithms. *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pp. 599–606.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996) Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, **26**(1):29–41.

Droste, S., Jansen, T., and Wegener, I. (2006) Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, **39**(4):525–544.

Earl, D. and Deem, M. W. (2004) Evolvability is a selectable trait. *PNAS*, **101**(32):11531–11536.

Echegoyen, C., Lozano, J. A., Santana, R., and Larrañaga, P. (2007) Exact bayesian network learning in estimation of distribution algorithms. *Procs IEEE CEC*, pp. 1051–1058.

Etxeberria, R. and Larrañaga, P. (1999) Global optimization using bayesian networks. *Proceedings of the Second Symposium on Artificial Intelligence. Adaptive Systems (CIMAF 99).*

Fodor, J. A. (1983) *The modularity of mind: an essay on faculty psychology.* MIT Press.

Fontanari, J. F. (1990) Generalization in a hopfield network. *Jornal de Physique*, **51**(21):2421–2430.

Forrest, S. and Mitchell, M. (1993a) Relative building-block fitness and the building block hypothesis. *foundations of genetic algorithms 2*, Whitley, D., ed.

——— (1993b) what makes a problem hard for a genetic algorithm? some anomalous results and their explanation. *Machine Learning*, **13**:285–319.

Frost, D. and Dechter, R. (1994) Dead-end driven learning. *Proceedings of the twelfth national conference on Artificial intelligence*, pp. 294–300, American Association for Artificial Intelligence, Menlo Park, CA, USA.

Gámez, J. A., Mateo, J. L., and Puerta, J. M. (2007) EDNA: Estimation of dependency networks algorithm. *Bio-inspired Modeling of Cognitive Tasks*, pp. 427–436, Springer.

Garibay, O., Garibay, I., and Wu, A. (2003) The modular genetic algorithm: exploiting regularities in the problem space. *Computer and Information Sciences - ISCIS 2003*, vol. LNCS 2869, pp. 584–591.

Geard, N. L. and Bullock, S. (2008) Group formation and social evolution: a computational model. *Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pp. 197–203, MIT Press.

Gimenez-Martinez, V. (2000) A modified hopfield auto-associative memory with improved capacity. *IEEE Transactions on Neural Networks*, **11**(4):867–878.

Goldberg, D. E. (1989) *Genetic Algorithm in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Grama, A., Gupta, A., Karypis, G., and Kumar, V. (2003) *Introduction to parallel computing*. 2nd edn., Addison-Wesley, Harlow, England.

Gross, T. and Blasius, B. (2008) Adaptive coevolutionary networks: a review. *Journal of The Royal Society Interface*, **5**(20):259–271.

Handa, H. (2007) The effectiveness of mutation operation in the case of estimation of distribution algorithms. *Biosystems*, **87**(2–3):243–251.

Harik, G. (1999) Linkage learning via probabilistic modeling in the ecga. *Tech. Rep. 99010*, University of Illinois, Urbana, IL.

Harik, G. R. (1994) Finding multiple solutions in problems of bounded difficulty. *Tech. Rep. 94002*, University of Illinois, Urbana, IL.

——— (1997) *Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms*. Ph.D. thesis, Department of Computer Science and Engineering, University of Michigan, Ann Arbor.

Harik, G. R., Lobo, F. G., and Goldberg, D. E. (1999) The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation*, **3**(4):287–297.

Hart, W. E. (1994) *Adaptive Global Optimization with Local Search*. Ph.D. thesis, University of California, San Diego.

Hartwell, L. H., Hopfield, J. J., Leibler, S., and Murray, A. W. (1999) From molecular to modular cell biology. *Nature*, **402**:C47–C52.

Hauschild, M., Pelikan, M., Lima, C. F., and Sastry, K. (2007) Analyzing probabilistic models in hierarchical boa on traps and spin glasses. *GECCO*, pp. 523–530, ACM Press.

Hebb, D. O. (1949) *The organization of behavior: a neuropsychological theory*. Wiley, New York.

Heckerman, D., Geiger, D., and Chickering, D. M. (1995) Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, **20**(3):197–243.

Henrion, M. (1986) Propagating uncertainty in bayesian networks by probabilistic logic sampling. *Proceedings of the 2nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-86)*, pp. 149–163, Elsevier, New York.

Higgs, P. G. (1996) Overlaps between RNA secondary structures. *Physical Review Letters*, **76**(4):704–707.

——— (2000) RNA secondary structure: physical and computational aspects. *Quarterly Reviews of Biophysics*, **33**(3):199–253.

Hillis, W. D. (1990) Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, **42**(1–3):228–234.

Hinton, G. E. and Nowlan, S. J. (1987) How learning can guide evolution. *Complex Systems*, **1**:495–502.

Hogeweg, P. (2007) From population dynamics to ecoinformatics: Ecosystems as multilevel information processing systems. *Ecological Informatics*, **2**(2):103–111.

Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

——— (2000) Building blocks, cohort genetic algorithms, and hyperplane-defined functions. *Evolutionary Computation*, **8(4)**:373–391.

Hopfield, J. J. (1982) Neural networks and physical systems with emergent collective computational abilities. *PNAS*, **79**:2554–2558.

Hopfield, J. J. and Tank, D. (1985) "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, **52**: 141–152.

——— (1986) Computing with neural circuits: A model. *Science*, **233**:625–623.

Houdayer, J. and Martin, O. C. (1999) Renormalization for discrete optimization. *Phys. Rev. Lett*, **83**:1030–1033.

Huang, C.-C. and Kusiak, A. (1998) Modularity in design of products and systems. *IEEE Transactions on Systems, Man and Cybernetics Part A*, **28**(1):66–77.

Husbands, P. and Mill, F. (1991) Simulated co-evolution as the mechanism for emergent planning and scheduling. *Procs. 4th International Conference on Genetic Algorithms*, Belew, L., R. & Booker, ed., pp. 264–270, Morgan Kaufmann.

Iclanzan, D. and Dumitrescu, D. (2007) Overcoming hierarchical difficulty by hill-climbing the building block structure. *GECCO*, pp. 1256–1263.

——— (2008) Going for the big fishes: discovering and combining large neutral and massively multimodal building-blocks with model based macro-mutation. *GECCO*, Ryan, C. and Keijzer, M., eds., pp. 423–430, ACM Press.

Ihmels, J., Friedlander, G., Bergmann, S., Sarig, O., Ziv, Y., and Barkai, N. (2002) Revealing modular organization in the yeast transcriptional network. *Nature Genetics*, **31**:370–377.

Irons, D. J. and Monk, N. A. M. (2007) Identifying dynamical modules from genetic regulatory systems: applications to the segment polarity network. *BMC Bioinformatics*, **8**:413.

Jansen, T. and Wegener, I. (2005) real royal road functions: where crossover is provably essential. *Discrete Applied Mathematics*, **149**(1-3):111–125.

Johnson, D. S. (1990) Local optimization and the traveling salesman problem. *Automata, Languages and Programming*, Paterson, M., ed., pp. 446–461, Springer.

Jones, T. (1995a) Crossover, macromutation, and population-based search. *Proceedings of the Sixth International Conference on Genetic Algorithms*, Morgan Kauffman.

——— (1995b) One operator, one landscape. *Tech. Rep. 95-02-025*, Santa Fe Institute, Santa Fe, NM 87501.

Kashtan, N. and Alon, U. (2005) Spontaneous evolution of modularity and network motifs. *PNAS*, **102**(39):13773–13778.

Kashtan, N., Noor, E., and Alon, U. (2007) Varying environments can speed up evolution. *PNAS*, **104**(34):13711–13716.

Kauffman, S., Macready, W. G., and Dickinson, E. (1994) Divide to coordinate: Coevolutionary problem solving. *Tech. Rep. 94-06-031*, Santa Fe Institute, Santa Fe, NM.

Kauffman, S. A. (1993) *The Origins of Order*. Oxford University Press USA.

Kauffman, S. A. and Johnsen, S. (1992) Coevolution of the edge of chaos: Coupled fitness landscapes, poised states, and coevolutionary avalanches. *Artifical Life II*, Langton, C., ed., Addison-Wesley, Reading, MA.

Khakhina, L. N. (1992) *Concepts of Symbiogenesis: Historical and Critical Study of the Research of Russian Botanists*. Yale University Press, editors: Lynn Margulis and Mark McMenamin. Translators: Stephanie Merkel and Robert Coalson.

Khare, V., Yao, X., Sendhoff, B., Jin, Y., and Wersing, H. (2005) Co-evolutionary modular neural networks for automatic problem decomposition. *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, pp. 2691–2698, IEEE Press.

Khare, V. R. (2006) *Automatic problem decomposition using co-evolution and modular neural networks*. Ph.D. thesis, University of Birmingham.

Khare, V. R. and Yao, X. (2004) Credit assignment among neurons in co-evolving populations. *Parallel Problem Solving from Nature VIII*, Yao, X. and Sendhoff, B., eds., pp. 882–891, Springer.

Kim, J. Y. and Kim, Y. K. (2005) Multileveled symbiotic evolutionary algorithm: Application to fms loading problems. *Applied Intelligence*, **22**(3):233–249.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983) Optimization by simulated annealing. *Science*, **220**(4598):671–680.

Klotz, T., Schubert, S., and Hoffmann, K. H. (1998) Coarse graining of a spin-glass state space. *Journal of Physics: Condensed Matter*, **10**(27):6127–6134.

Koza, J. R. (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

Krasnogor, N. and Smith, J. (2005) A tutorial for competent memetic algorithms: model, taxonomy and design issues. *IEEE Transactions on Evolutionary Computation*, **9**(5):474–488.

Krause, A. E., Frank, K. A., Mason, D. M., Ulanowicz, R. E., and Taylor, W. W. (2003) Compartments revealed in food-web structure. *Nature*, **426**:282–285.

Laan, J. D. V. D. and Hogeweg, P. (1995) Predator-prey coevolution: Interactions across different timescales. *Proceedings of the Royal Society of London. Series B*, **259**(1354):35–42.

Larrañaga, P., Etxeberria, R., Lozano, J. A., and na, J. P. (1999) Optimization by learning and simulation of bayesian and gaussian networks. *Tech. Rep. EHU-KZAA-IK-4/99*, University of the Basque Country.

Larrañaga, P., Etxeberria, R., Lozano, J., and Peña, J. (2000) Combinatorial optimization by learning and simulation of bayesian networks. *Conference in Uncertainty on Artificial Intelligence*, pp. 343–352, Stanford, CA.

Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R. H., and Kuijpers, C. M. (1996) Structure learning of bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **18**(9):912–926.

Larsen, R. J. and Marx, M. L. (2006) *An introduction to mathematical statistics and its applications*. Fourth edn., Pearson Prentice Hall, Upper Saddle River, NJ.

Lavers, C. (2000) *Why Elephants have Big Ears*. Phoenix, London.

Lenaerts, T., Defaweux, A., Beyens, P., and Manderick, B. (2001) Transitions in a simple evolutionary model. *Proceedings of the 6th European Conference on Artificial Life (ECAL)*, Springer.

Lenaerts, T., Defaweux, A., van Remortel, P., and Manderick, B. (2003) Modelling artificial multi-level selection. *AAAI Spring symposium on computational synthesis.*

Levin, S. A. (1998) Ecosystems and the biosphere as complex adaptive systems. *Ecosystems*, **1**(5):431–436.

——— (1999) *Fragile Dominion: Complexity and the Commons.* Perseus, Cambridge, MA.

Lewis, M. (2009) *An Investigation into the Evolution of Relationships Between Species in an Ecosystem.* Master's thesis, Electronics and Computer Science, University of Southampton.

Li, Z. and Goodman, E. D. (2007) Learning building block structure from crossover failure. *Proceedings of the Genetic and Evolutionary Computation Conference*, Thierens, D. and Lipson, H., eds., pp. 1280–1287, ACM Press.

Lima, C. F., Pelikan, M., Sastry, K., Butz, M., Goldberg, D. E., and Lobo, F. G. (2006) Substructural neighborhoods for local search in the bayesian optimization algorithm. *Parallel Problem Solving from Nature IX*, pp. 232–241, Springer.

Lima, C. F., Sastry, K., Goldberg, D. E., and Lobo, F. G. (2005) Combining competent crossover and mutation operators: a probabilistic model building approach. *GECCO*, et al, H. B., ed., pp. 735–742, ACM Press.

Linsker, R. (1988) Self-organization in a perceptual network. *Computer*, **21**(3):105–117.

Lipson, H. (2007) Principles of modularity, regularity, and hierarchy for scalable systems. *Journal of Biological Physics and Chemistry*, **7**(4):125–128.

Lipson, H., Pollack, J. B., and Suh, N. P. (2001) Promoting modularity in evolutionary design. *Proceedings of DETC ASME Design Engineering Technical Conferences.*

——— (2002) On the origin of modular variation. *Evolution*, **56**(8):1549–1556.

Looks, M. (2006) *Competent Program Evolution.* Ph.D. thesis, Washington University.

Lourenco, H. R., Martin, O., and Stützle, T. (2003) *Handbook of Metaheuristics*, chap. Iterated Local Search, pp. 321–353. Springer.

Lozano, J. A., Larrañaga, P., Inza, I., and Bengoetxea, E. (eds.) (2006) *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms.* Springer.

Luke, S. (2009) *Essentials of Metaheuristics.* Available at http://cs.gmu.edu/∼sean/book/metaheuristics/.

Mahdavi, K., Harman, M., and Hierons, R. M. (2003) A multiple hill climbing approach to software module clustering. *Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference*, pp. 315–324.

Mahfoud, S. (1992) Crowding and preselection revisited. *Parallel problem solving from nature 2*, pp. 27–36.

Margulis, L. (1998) *The Symbiotic Planet*. Phoenix, London.

Margulis, L., Dolan, M. F., and Guerrero, R. (2000) The chimeric eukaryote: Origin of the nucleus from the karyomastigontin amitochondriate protists. *PNAS*, **97**(13):6954–6959.

Martin, O. C. and Otto, S. W. (1996) Combining simulated annealing with local search heuristics. *Annals of Operations Research*, **63**(1):57–75.

Martín González, A. M., Dalsgaard, B., and Olesen, J. M. (2009) Centrality measures and the importance of generalist species in pollination networks. *Ecological Complexity*.

Maynard Smith, J. and Szathmáry, E. (1995) *The Major Transitions in Evolution*. Oxford University Press.

Mayr, E. (2001) *What Evolution is*. Phoenix, London.

McEliece, R., Posner, E., Rodemich, E., and Venkatesh, S. (1987) The capacity of the hopfield associative memory. *IEEE Transactions on Information Theory*, **33**:461–482.

McGregor, S. and Harvey, I. (2005) Embracing plagiarism: Theoretical, biological and empirical justification for copy operators in genetic optimisation. *Genetic Programming and Evolvable Machines*, **6**:407–420.

Middleton, A. A. (1995) Numerical results for the ground-state interface in a random medium. *Physical Review E*, **52**(4):R3337–R3340.

Mills, R. and Watson, R. A. (2005) Genetic assimilation and canalisation in the Baldwin effect. *Proceedings of the VIIIth European Conference on Artificial Life*, Capcarrère, M. S., Freitas, A. A., Bentley, P. J., Johnson, C. G., and Timmis, J., eds., pp. 353–362, Springer.

——— (2006) On crossing fitness valleys with the Baldwin effect. *Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems (ALifeX)*, Rocha, L. M., Yaeger, L. S., Bedau, M. A., Floreano, D., Goldstone, R. L., and Vespignani, A., eds., pp. 493–499, MIT Press.

——— (2007a) Symbiosis, synergy and modularity: Introducing the reciprocal synergy symbiosis algorithm. *Proceedings of the IXth European Conference on Artificial Life*, e Costa, F. A., ed., pp. 1192–1201, Springer.

——— (2007b) Variable discrimination of crossover versus mutation using parameterized modular structure. *Proceedings of the Genetic and Evolutionary Computation Conference*, Thierens, D. and Lipson, H., eds., pp. 1312–1319, ACM Press.

——— (2008) Adaptive units of selection can evolve complexes that are provably unevolvable under fixed units of selection (abstract). *11th International Conference on the Simulation and Synthesis of Living Systems*, Bullock, S., Noble, J., Watson, R. A., and Bedau, M. A., eds., p. 785, MIT Press.

——— (2009) Symbiosis enables the evolution of rare complexes in structured environments. *Proceedings of the 10th European Conference on Artificial Life*, Kampis, G. and Szathmary, E., eds., Springer.

Mitchell, M. (1996) *An Introduction to Genetic Algorithms.* MIT Press.

Mitchell, M., Holland, J. H., and Forrest, S. (1994) When will a genetic algorithm outperform hill climbing? *Advances in Neural Information Processing Systems*, Cowan, J. D., Tesauro, G., and Alspector, J., eds., vol. 6, pp. 51–58, Morgan Kaufmann Publishers, Inc.

Morgan, S. R. and Higgs, P. G. (1998) Barrier heights between ground states in a model of RNA secondary structure. *Journal of Physics A: Mathematical and General*, **31**(14):3153–3170.

Moriarty, D. E. and Miikkulainen, R. (1998) Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, **5**(4):373–399.

Motwani, R. and Raghavan, P. (1995) *Randomized algorithms.* Cambridge University Press.

Mühlenbein, H. (1992) How genetic algorithms really work: Mutation and hillclimbing. *Parallel Problem Solving from Nature 2*, Männer, R. and Manderick, B., eds., pp. 15–26, Elsevier.

Mühlenbein, H. (1997) The equation for response to selection and its use for prediction. *Evolutionary Computation*, **5**(3):303–346.

Mühlenbein, H. and Mahnig, T. (1999) FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, **7**:353–376.

Mühlenbein, H., Mahnig, T., and Rodriguez, A. (1999) Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, **5**:215–247.

Mühlenbein, H. and Paaß, G. (1996) From recombination of genes to the estimation of distributions i. binary parameters. *Parallel Problem Solving from Nature PPSN IV*, vol. LNCS 1141, pp. 178–187.

Myers, J. W., Laskey, K. B., and De Jong, K. A. (1999) Learning bayesian networks from incomplete data using evolutionary algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference*, Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., eds., Morgan Kaufmann.

Newman, M. E. J. (2006) Modularity and community structure in networks. *PNAS*, **103**(23):8577–8582.

Newman, M. E. J. and Girvan, M. (2004) Finding and evaluating community structure in networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, **69**(2).

Odling-Smee, F. J. (1995) Niche construction, genetic evolution and cultural change. *Behavioural Processes*, **35**(1–3):195–205.

Olesen, J. M., Bascompte, J., Dupont, Y. L., and Jordano, P. (2007) The modularity of pollination networks. *Proceedings of the National Academy of Sciences*, **104**(50):19891–19896.

Page, S. E., Sander, L. M., and Schneider-Mizell, C. M. (2007) Conformity and dissonance in generalized voter models. *Journal of Statistical Physics*, **128**(6):1279–1287.

Pagie, L. and Hogeweg, P. (1998) Evolutionary consequences of coevolving targets. *Evolutionary Computation*, **5**(4):401–418.

Parker, G. B. and Blumenthal, H. J. (2003) Comparison of sampling sizes for the coevolution of cooperative agents. *Congress on Evolutionary Computation*, pp. 536–543.

Parter, M., Kashtan, N., and Alon, U. (2008) Facilitated variation: How evolution learns from past environments to generalize to new environments. *PLoS Computational Biology*, **4**(11):e1000206.

Pelikan, M. (2002) *Bayesian optimisation algorithm: from single level to hierarchy*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

Pelikan, M. and Goldberg, D. E. (2000) Hierarchical problem solving and the bayesian optimization algorithm. *GECCO*, pp. 267–274.

——— (2001a) bayesian optimization algorithm, decision graphs and occams razor. *Proceedings of the Genetic and Evolutionary Computation Conference*, Spector, L., Goodman, E. D., and Wu, A., eds., pp. 519–526, Morgan Kaufmann, San Fransisco, CA.

——— (2001b) Escaping hierarchical traps with competent genetic algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference*, Spector, L., Goodman, E. D., and Wu, A., eds., pp. 511–518, Morgan Kaufmann, San Fransisco, CA.

——— (2003) Hierarchical BOA solves Ising spin glasses and MAXSAT. *GECCO*, pp. 1271–1282, Springer.

Pelikan, M., Goldberg, D. E., and Cantú Paz, E. (1999) BOA: The Bayesian optimization algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference*, Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., eds., pp. 525–532, Morgan Kaufmann Publishers, San Fransisco, CA, Orlando, FL.

Pelikan, M., Goldberg, D. E., and Lobo, F. (2002) A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, **21**(1):5–20, also IlliGAL Report No. 99018.

Pelikan, M., Goldberg, D. E., and Tsutsui, S. (2003) Hierarchical bayesian optimisation algorithm: Toward a new generation of evolutionary algorithms. *SICE Annual Conference*, pp. 2738–2743.

Pelikan, M. and Hartmann, A. K. (2007) Obtaining ground states of Ising spin glasses via optimizing bonds instead of spins. *GECCO*, p. 628, ACM Press.

Pelikan, M. and Mühlenbein, H. (1999) The bivariate marginal distribution algorithm. *Advances in Soft Computing — Engineering Design and Manufacturing*, Roy, R., Furuhashi, T., and Chawdhry, P. K., eds., pp. 521–535, Springer.

Pelikan, M., Sastry, K., Butz, M. V., and Goldberg, D. E. (2006a) Hierarchical BOA on random decomposable problems. *GECCO*, Keijzer, M. and Cattolico, M., eds., pp. 431–432, ACM Press.

Pelikan, M., Sastry, K., and Cantú-Paz, E. (eds.) (2006b) *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*. Springer.

Pelikan, M., Sastry, K., and Goldberg, D. E. (2008) iboa: The incremental bayesian optimization algorithm. *Tech. Rep. 2008002*, University of Missouri, St. Mouis, MO.

Penn, A. and Harvey, I. (2004) The role of non-genetic change in the heritability, variation, and response to selection of artificially selected ecosystems. *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Life (ALife IX)*, pp. 352–357, MIT Press.

Penn, A. S. (2002) Towards a quantitative analysis of individuality and its maintenance. *Procs SAB*, pp. 385–386, MIT Press.

Philemotte, C. and Bersini, H. (2006) How an optimal observer can collapse the search space. *GECCO*, pp. 1273–1280, ACM Press.

——— (2007) A gestalt genetic algorithm: less details for better search. *GECCO*, pp. 1328–1334, ACM Press.

Poderoso, F. C. and Fontanari, J. F. (2007) Model ecosystem with variable interspecies interactions. *Journal of Physics A*, **40**: 8723–8738.

Potter, M. A. (1997) *The design and analysis of a computational model of cooperative coevolution.* Ph.D. thesis, George Mason University, Fairfax, VA, USA.

Potter, M. A. and De Jong, K. A. (1994) A cooperative coevolutionary approach to function optimization. *Parallel Problem Solving from Nature III*, Goos, G., Hartmanis, J., and van Leeuwen, J., eds., pp. 249–257, Springer.

——— (1995) Evolving neural networks with collaborative species. *Procs 1995 Summer Computer Simulation Conference*, Oren, T. I. and Birta, L., eds., pp. 340–345, Society for Computer Simulation, San Diego, CA.

——— (2000) Cooperative coevolution: an architecture for evolving coadapted subcomponents. *Evolutionary Computation*, **8**(1): 1–29.

Powers, S. T., Mills, R., Penn, A. S., and Watson, R. A. (2009) Social environment construction provides an adaptive explanation for new levels of individuality. *Levels of Selection and Individuality in Evolution: Conceptual Issues and the Role of Artificial Life Models. Workshop at ECAL*, pp. 18–21.

Qasem, M. and Prügel-Bennett, A. (2008) Complexity of max-SAT using stochastic algorithms. *Genetic And Evolutionary Computation Conference*, Ryan, C. and Keijzer, M., eds., pp. 615–616, ACM Press.

——— (2009) Improving performance in combinatorial optimisation using averaging and clustering. *EvoCOP '09: Evolutionary Computation in Combinatorial Optimization*, Cotta, C. and Cowling, P., eds., pp. 180–191, Springer.

Queller, D. C. (1997) Cooperators since life began. *The Quarterly Review of Biology*, **72**(2):184–188.

Ramirez, A. P. (1994) Strongly geometrically frustrated magnets. *Annual Review of Materials Science*, **24**:453–480.

Ridley, M. (2004) *Evolution.* Third edn., Blackwell.

Santana, R. (2003) a markov network based factorized distribution algorithm for optimization. *ECML*, vol. 2837, pp. 337–348, Springer, Berlin.

——— (2005) Estimation of distribution algorithms with kikuchi approximations. *Evolutionary Computation*, **13**(1):67–97.

Santana, R., Larrañaga, P., and Lozano, J. A. (2005) Interactions and dependencies in estimation of distribution algorithms. *Procs IEEE CEC*, pp. 1418–1425.

——— (2006) Mixtures of kikuchi approximations. *ECML*, pp. 365–376, Springer.

Santos, F. C., Pacheco, J. M., and Lenaerts, T. (2006) Cooperation prevails when individuals adjust their social ties. *PLoS Computational Biology*, **2**:e140.

Sastry, K. and Goldberg, D. E. (2004) Designing competent mutation operators via probabilistic model building of neighborhoods. *GECCO*, pp. 114–125, Springer.

Schlosser, G. (2002) Modularity and the units of evolution. *Theory in Biosciences*, **121**(1):1–80.

Segré, D., Ben-Eli, D., and Lancet, D. (2000) Compositional genomes: Prebiotic information transfer in mutually catalytic noncovalent assemblies. *PNAS*, **97**(8):4112–4117.

Shakya, S. and McCall, J. (2007) Optimization by estimation of distribution with deum framework based on markov random fields. *International Journal of Automation and Computing*, **4**(3):262–272.

Shakya, S. K. (2006) *DEUM: A framework for an Estimation of Distribution Algorithm based on Markov Random Fields.* Ph.D. thesis, Robert Gordon University, Aberdeen, UK.

Shakya, S. K., McCall, J. A., and Brown, D. F. (2004) Updating the probability vector using mrf techniques for a univariate eda. *STAIRS*, pp. 15–25, IOS press.

——— (2006) Solving the Ising spin glass problem using a bivariate EDA based on Markov random fields. *IEEE CEC*, pp. 908–915.

Shi, Y.-j., Teng, H.-f., and Li, Z.-q. (2005) Cooperative co-evolutionary differential evolution for function optimization. *Procs. First International Conference on Natural Computation*, pp. 1080–1088, Springer.

Shih, C. J. and Yang, Y. C. (2002) Generalized hopfield network based structural optimization using sequential unconstrained minimization technique with additional penalty strategy. *Advances in Engineering Software*, **33**(7–10):721–729.

Simon, H. A. (1969) *The sciences of the Artificial.* Third (1996) edn., MIT Press.

——— (2002) Near decomposability and the speed of evolution. *Industrial and Corporate Change*, **11**(3):587–599.

Singh, G. and Deb, K. (2006) Comparison of multi-modal optimization algorithms based on evolutionary algorithms. *Procs GECCO*, pp. 1305–1312, ACM Press.

Smith, M. J. (1997) *Application-Specific Integrated Circuits.* Addison-Wesley, Reading, MA.

Snel, B. and Huynen, M. A. (2004) Quantifying modularity in the evolution of biomolecular systems. *Genome research*, **14(3)**:391–397.

Soetaert, K. and Herman, P. M. (2009) *A Practical Guide to Ecological Modelling*, chap. Multiple Time Scales and Equilibrium Processes, pp. 257–272. Springer.

Sofge, D., De Jong, K., and Schultz, A. (2002) A blended population approach to cooperative coevolution for decomposition of complex problems. *Procs CEC*, pp. 413–418.

Solé, R. V., Alonso, D., and McKane, A. (2000) Scaling in a network model of a multispecies ecosystem. *Physica A*, **286**(1–2): 337–344.

Spears, W. M. (1993) Crossover or mutation? *Foundations of Genetic Algorithms 2*, Whitley, D., ed., pp. 221–237, Morgan Kaufmann.

Sporns, O. (2006) Small-world connectivity, motif composition, and complexity of fractal neuronal connections. *Biosystems*, **85**(1): 55–64.

Sterelny, K. (2004) Symbiosis, evolvability and modularity. *Modularity in development and evolution*, Schlosser, G. and Wagner, G., eds., University of Chicago Press.

Sun, J. and Deem, M. W. (2007) Spontaneous emergence of modularity in a model of evolving individuals. *Phys Rev Lett*, **99**:228107.

Sutherland, J. P. (1974) Multiple stable points in natural communities. *The American Naturalist*, **108**(964):859–873.

Syswerda, G. (1993) Simulated crossover in genetic algorithms. *Foundations of Genetic Algorithms 2*, Whitley, L. D., ed., pp. 239–255, Morgan Kauffmann, San Mateo, CA.

Thompson, J. N. (1988) Variation in interspecific interactions. *Annual Review of Ecology and Systematics*, **19**:65–87.

——— (1994) *The Coevolutionary Process*. University of Chicago Press.

Toussaint, M. (2003) The structure of evolutionary exploration: On crossover, buildings blocks, and estimation-of-distribution algorithms. *GECCO*, pp. 1444–1456, Springer.

——— (2005) Compact genetic codes as a search strategy of evolutionary processes. *Foundations of Genetic Algorithms 8*, Wright, A. H., Vose, M. D., Jong, K. A. D., and Schmitt, L. M., eds., no. 3469 in LNCS, pp. 75–94.

Toussaint, M. and von Seelen, W. (2007) Complex adaptation and system structure. *Biosystems*, **90**(3):769–782.

van den Bergh, F. and Engelbrecht, A. P. (2004) A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, **8**(3):225–239.

van der Horst, J., Noble, J., and Tantall, A. (2009) Robustness of market-based task allocation in a distributed satellite system. *Proceedings of the 10th European Conference on Artificial Life*, Kampis, G. and Szathmary, E., eds., Springer.

Van Segbroeck, S., Santos, F. C., Lenaerts, T., and Pacheco, J. M. (2009) Reacting differently to adverse ties promotes cooperation in social networks. *Physical Review Letters*, **102**(5):058105.

Vilar, J. M. G. (2006) Modularizing gene regulation. *Molecular Systems Biology*, **2**:16.

Vinyals, M., Rodríguez-Aguilar, J. A., and Cerquides, J. (forthcoming) A survey on sensor networks from a multiagent perspective. *The Computer Journal*.

Voss, M. S. and Foley, C. M. (1999) Evolutionary algorithm for structural optimization. *Procs GECCO*, Banzhaf, W., Daida, J. M., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M. J., and Smith, R. E., eds., pp. 678–685, Morgan Kauffman.

Wade, M. J. and Goodnight, C. J. (1998) Perspective: The theories of fisher and wright in the context of metapopulations: When nature does many small experiments. *Evolution*, **52**(6):1537–1553.

Wagner, G. P. (1996) Homologues, natural kinds and the evolution of modularity. *American Zoologist*, **36**(1):36–43.

Wagner, G. P. and Altenberg, L. (1996) Complex adaptations and the evolution of evolvability. *Evolution*, **50**(3):967–976.

Walker, J. A. and Miller, J. F. (2005) Investigating the performance of module acquisition in cartesian genetic programming. *Procs GECCO*, pp. 1649–1656, ACM Press.

——— (2008) The automatic acquisition, evolution and reuse of modules in cartesian genetic programming. *IEEE Transactions on Evolutionary Computation*, **12**(4):397–417.

Wallin, D., Ryan, C., and Azad, R. M. A. (2005) Symbiogenetic coevolution. *Proceedings of the Congress on Evolutionary Computation*, pp. 1613–1620.

Watson, R. A. (2004) A simple two-module problem to exemplify building-block assembly under crossover. *Parallel Problem Solving from Nature VIII*, pp. 161–171.

——— (2005) On the unit of selection in sexual populations. *Proceedings of the VIIIth European Conference on Artificial Life*, Capcarrère, M. S., Freitas, A. A., Bentley, P. J., Johnson, C. G., and Timmis, J., eds., pp. 895–905, Springer.

——— (2006) *Compositional Evolution: the impact of sex, symbiosis, and modularity on the gradualist framework of evolution*. MIT Press.

Watson, R. A., Buckley, C., and Mills, R. (2009a) The effect of Hebbian learning on optimisation in Hopfield networks. *Tech. rep.*, ECS, University of Southampton.

——— (2009b) Global efficiency in networks of selfish components: Emergent associative memory at the system scale. *Tech. rep.*, ECS, University of Southampton.

——— (2010) Enhanced resolution of constraints in ‘self-modelling’ complex adaptive systems. *Complexity*, (in revision).

Watson, R. A., Hornby, G. S., and Pollack, J. B. (1998) Modelling building-block interdependency. *parallel problem solving from nature V*, pp. 97–106.

Watson, R. A. and Jansen, T. (2007) A building-block royal road where crossover is provably essential. *Proceedings of the Genetic and Evolutionary Computation Conference*, Thierens, D. and Lipson, H., eds., pp. 1452–1459, ACM Press.

Watson, R. A., Mills, R., Powers, S., and Penn, A. (2008) Can individual selection favour significant higher-level selection? (abstract). *The Eleventh International Conference on the Simulation and Synthesis of Living Systems (Alife XI)*, Bullock, S., Noble, J., Watson, R. A., and Bedau, M., eds., p. 818.

Watson, R. A., Palmius, N., Mills, R., Powers, S., and Penn, A. (2009c) Can selfish symbioses effect higher-level selection? *Proceedings of the 10th European Conference on Artificial Life*.

Watson, R. A. and Pollack, J. B. (2000) Symbiotic combination as an alternative to sexual recombination in genetic algorithms. *PPSN VI*, et al, M. S., ed., LNCS 1917, pp. 425–434.

——— (2003) A computational model of symbiotic composition in evolutionary transitions. *Biosystems*, **69**(2-3):187–209.

——— (2005) Modular interdependency in complex dynamical systems. *Artificial Life*, **11**(4):445–457.

Watts, D. J. and Strogatz, S. H. (1998) Collective dynamics of 'small world' networks. *Nature*, **393**:440–442.

Weinreich, D. M., Watson, R. A., and Chao, L. (2005) Perspective: sign epistasis and genetic costraint on evolutionary trajectories. *Evolution*, **59**(6):1165–1174.

Weiss, M. A. (1997) *Data structure and algorithm analysis in C*. Addison-Wesley.

Whitlock, M. C., Phillips, P. C., Moore, F. B., and Tonsor, S. J. (1995) Multiple fitness peaks and epistasis. *Annual Review of Ecology and Systematics*, **26**:601–629.

Wiegand, R. P., Liles, W. C., and De Jong, K. A. (2001) An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. *Procs Genetic and Evolutionary Computation Conference*, pp. 1235–1245, Morgan Kauffman.

Wilkins, J. and Godfrey-Smith, P. (2009) Adaptationism and the adaptive landscape. *Biology and Philosophy*, **24**(2):199–214.

Williams, H. T. (2006) *Homeostatic adaptive networks*. Ph.D. thesis, University of Leeds.

Wimsatt, W. C. and Schank, J. C. (2004) Generative entrenchment, modularity and evolvability: When genic selection meets the whole organism. *Modularity in Development and Evolution*, Schlosser, G. and Wagner, G., eds., pp. 359–394, University of Chicago Press.

Worgan, S. and Mills, R. (2008) Initial modelling of the alternative phenotypes hypothesis. *Proceedings of the Eleventh International Conference on Artificial Life*, Bullock, S., Noble, J., Watson, R. A., and Bedau, M. A., eds., pp. 717–724, MIT Press.

Wright, S. (1931) Evolution in mendelian populations. *Genetics*, **16**(3):97–159.

——— (1935) Evolution in populations in approximate equilibrium. *Journal of Genetics*, **30**(2):257–266.

Yang, Z., Tang, K., and Yao, X. (2008) Large scale evolutionary optimization using cooperative coevolution. *Information Sciences*, **178**(15):2985–2999.

Yong, C. H. and Miikkulainen, R. (2001) Cooperative coevolution of multi-agent systems. *Tech. Rep. AI-01-287*, University of Texas at Austin.

Yu, T.-L. (2007) *A Matrix Approach for Finding Extrema: Problems with Modularity, Hierarchy, and Overlap*. Ph.D. thesis, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, IL 61801.

Yu, T.-L. and Goldberg, D. E. (2006) Conquering hierarchical difficulty by explicit chunking: Substructural chromosome compression. *GECCO*, pp. 1385–1392, ACM Press.

Yu, T.-L., Goldberg, D. E., Yassine, A., and Chen, Y.-P. (2003) Genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm. *GECCO*, pp. 1620–1621, Springer.

Zhang, Q., Sun, J., Tsang, E., and Ford, J. (2004) Hybrid estimation of distribution algorithm for global optimization. *Engineering Computations*, **21**(1):91–107.